

## Systemunterlagen

### Literatur

1. Günter Schmitt  
Mikrocomputertechnik mit dem 16-Bit-Prozessor 8086  
Oldenbourg 1986 - MIC Grundlagen und Assembler-Programmierung
2. Hans-Peter Messmer  
PC-Hardwarebuch  
Addison-Wesley 2000 Hardware - Standardwerk
3. Michael Tischer  
PC Intern 5 - Systemprogrammierung  
DATA Becker 1995 Standardwerk der Systemprogrammierung
4. Oliver Müller  
Assembler - Referenz  
Franzis 2000 insbesondere Assembler unter Windows und Einbindung in C / C++
5. E.-W. Dieterich  
Assembler - Grundlagen der PC-Programmierung  
Oldenbourg 2000 insbesondere Verbindung von Assembler und C / C++
6. Brors Isa  
Maschinensprache des IBM-PC/AT  
Hüthig, 1992 - Assembler, Systemprogrammierung, nötige an Hardware
7. Thomas Little  
Das PC-Buch  
Systema Verlag, 1992 - Hardware, Systemprogrammierung, wenig Assemblerprogrammierung
8. Wolfgang Link  
Assembler Programmierung (Einführung unter MS-DOS)  
Franzis-Verlag 1992 - Assembler-Programmierung
9. Holger Schökel  
Maschinensprache Einsteiger  
DATA Becker 1992 - insbesondere auch TSR- und Treiberprogrammierung
10. H. Weber  
Mikrorechnertechnik - Die Intel Mikroprozessorfamilie  
R. Oldenburg München Wien 1994 Hardware ( nicht nur PC-spezifisch ) und Assemblerprogrammierung
11. G. Schmitt  
C++ Kurs – technisch orientiert  
R. Oldenbourg Verlag München Wien 1999 insbesondere Hardware-Zugriffe unter DOS
12. F. Bollow, K.-P. Köhn  
PC-System-Programmierung  
VDE-Verlag Berlin Offenbach 1995 in Assembler unter DOS
13. T. Langenkamp  
Hardware- und System-Programmierung  
te-wi Verlag 1994 Assembler und C, sehr ausführlich VGA-Programmierung

14. BORLAND C++ 3.0  
    Programmier-Handbuch ( u. Benutzer-Handbuch )  
    Borland 1992  
        insbesondere auch Inline-Assemblierung
15. Turbo Assembler 3.0  
    Benutzerhandbuch  
    Borland 1992  
        insbesondere Schnittstelle zu Borland C++
16. Frank van Gilluwe  
    The undocumented PC  
    Addison-Wesley 1994  
        alles über BIOS / DOS und Hardware
17. Tom Hogan  
    Die PC-Referenz für Programmierer  
    Sythema Verlag 1992  
        reines Nachschlagewerk

### Programmiermodell des INTEL 8086-Prozessors ( Real Mode )

enthält die Register mit ihren symbolischen Namen, die für den Programmierer erreichbar sind.

| Bit-Nummer | 15 | 8 | 7  | 0 |  |                                                                                                                                                                                                                                                                |
|------------|----|---|----|---|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>AX</b>  | AH |   | AL |   |  | <b>Arbeitsregister</b> allg. Arbeitsregister 16 Bit breit<br>auch als zwei 8-Bit-Register verwendbar<br>H -> High höherwertiges<br>L-> Low niederwertiges Byte                                                                                                 |
|            | BH |   | BL |   |  |                                                                                                                                                                                                                                                                |
|            | CH |   | CL |   |  |                                                                                                                                                                                                                                                                |
|            | DH |   | DL |   |  |                                                                                                                                                                                                                                                                |
| <b>BX</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>CX</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>DX</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>SI</b>  |    |   |    |   |  | <b>Offsetregister</b><br>Indexregister SI -> Source Index ( Quellindex )<br><br>DI -> Destination Index (Zielindex )                                                                                                                                           |
| <b>DI</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>SP</b>  |    |   |    |   |  | <b>Stapelzeigerregister</b> SP-> Stackpointer zeigt auf den aktuellen<br>Eintrag im Stack ( Stapel )<br>BP -> Basepointer                                                                                                                                      |
| <b>BP</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>CS</b>  |    |   |    |   |  | <b>Segmentregister</b> CS-> Codesegment, zeigt auf Speicher-<br>segment mit aktuellem Programmcode<br>DS->Datensegment, zeigt auf Daten<br><br>ES->Extrasegment ( 2.Datensegment )<br><br>SS->Stacksegment, zeigt auf Stapelsegment<br>für Zwischenspeicherung |
| <b>DS</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>ES</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>SS</b>  |    |   |    |   |  |                                                                                                                                                                                                                                                                |
| <b>IP</b>  |    |   |    |   |  | <b>Befehlszeiger</b> IP-> Instruction Pointer zeigt auf die Speicher-<br>adresse mit dem nächsten auszuführenden<br>Befehl                                                                                                                                     |
| <b>F</b>   |    |   |    |   |  | <b>Prozessorstatusregister</b> F -> Flagregister , die einzelnen Flags<br>(Bits) weisen auf wichtige interne<br>Prozessorzustände hin.                                                                                                                         |

### Flagregister (Statusregister) F :

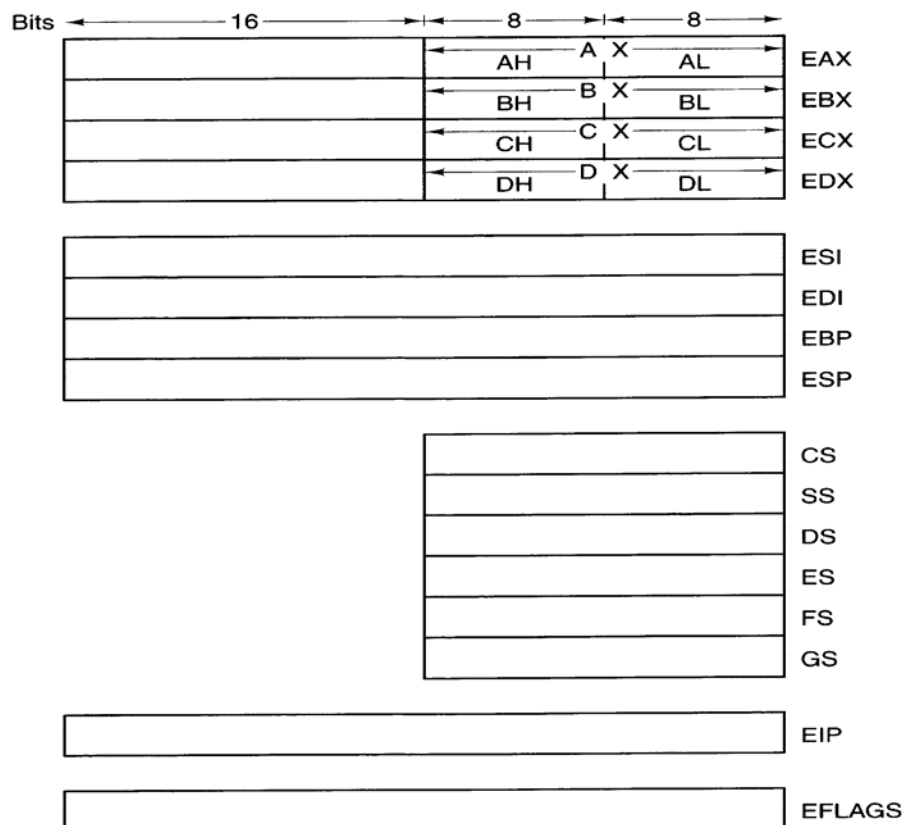
| 15 | 14 | 13 | 12 | 11       | 10       | 9        | 8        | 7        | 6        | 5 | 4        | 3 | 2        | 1 | 0        |
|----|----|----|----|----------|----------|----------|----------|----------|----------|---|----------|---|----------|---|----------|
| 0  | NT | IO | PL | <b>O</b> | <b>D</b> | <b>I</b> | <b>T</b> | <b>S</b> | <b>Z</b> | 0 | <b>A</b> | 0 | <b>P</b> | 1 | <b>C</b> |

| Kurzbez. | Bezeichnung | Debug-Bez.            | Bedeutung                                                                                                                     |
|----------|-------------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------|
|          |             | 1 0                   |                                                                                                                               |
| <b>C</b> | Carry       | CY / NC               | C zeigt einen Übertrag aus der höchstwertigen Stelle an wird benötigt bei arithmetischen und logischen Operationen            |
| <b>P</b> | Parity      | PE / PO<br>even / odd | Im niederwertigen Byte des Ergebnisses ist die Anzahl der auf 1 stehenden Bits gerade (even) (keine Ergänzung !! nur Anzeige) |

|          |                 |         |                                                                                                           |
|----------|-----------------|---------|-----------------------------------------------------------------------------------------------------------|
| <b>A</b> | Auxiliary Carry | AC / NA | zeigt einen Übertrag von Bit 3 nach Bit 4 an<br>benötigt für Dezimalkorrektur beim Rechnen mit BCD-Zahlen |
| <b>Z</b> | Zero            | ZR / NZ | zeigt an, ob das Ergebnis einer Operation 0 ist                                                           |
| <b>S</b> | Sign            | NG / PL | zeigt das Vorzeichen eines Ergebnisses an ( höchstwertiges Bit )                                          |
| <b>O</b> | Overflow        | OV / NV | Überlauf, d.h. Vorzeichenumkehr, Überschreiten des Zahlenbereiches vorzeichenbehalteter Dualzahlen        |

### Steuerflags

|          |                  |         |                                                                                                                    |
|----------|------------------|---------|--------------------------------------------------------------------------------------------------------------------|
| <b>D</b> | Direction        | DN / UP | gibt bei Stringoperationen die Indexrichtung (inkrementieren oder dekrementieren) an                               |
| <b>I</b> | Interrupt Enable | EI / DI | maskierbare externe Interrupts werden zugelassen (INTR-Eingang) / gesperrt                                         |
| <b>T</b> | Trap             |         | nach Ausführung eines Maschinenbefehls wird ein Interrupt ausgelöst, um Programme zu testen ( Einzelschrittmodus ) |



**Die Register des Pentium II** ( Register ab 80386 )

## Zusammenfassung der DEBUG-Befehle

| Befehl     | Semantik                                  | Format                           |
|------------|-------------------------------------------|----------------------------------|
| Assemble   | Assembleranweisungen umwandeln            | <b>a</b> [Adresse]               |
| Compare    | Speicherbereiche miteinander vergleichen  | <b>c</b> Bereich Adr             |
| Dump       | Speicherinhalt anzeigen                   | <b>d</b> [Adr] oder d [Bereich]  |
| Enter      | Speicherinhalt ändern / eingeben          | <b>e</b> Adr Liste               |
| Fill       | Speicherbereich mit Muster füllen         | <b>f</b> Bereich Muster          |
| Go         | Pgrammausführung mit evtl. Breakpoints    | <b>g</b> [=Adr][Adrl][Adr2]...   |
| Hexarithm. | hexadez. Addition und Subtraktion         | <b>h</b> Wert Wert               |
| Input      | Byte von I/O-Adresse lesen                | <b>i</b> Portadresse             |
| Load       | Dateien oder abs. Diskettensektoren laden | <b>l</b> [Adr [Laufw Sek. Sek.]] |
| Move       | Speicherbereich übertragen                | <b>m</b> Bereich Adr             |
| Name       | Dateien und Parameter definieren          | <b>n</b> [d:] [Pfad]Name[.erw]   |
| Output     | Byte an I/O-Adresse senden                | <b>o</b> Portadresse Byte        |
| Proceed    | Stopp bei nächster Instruktion            | <b>p</b> [=Adr] [Wert]           |
| Quit       | DEBUG-Prog. verlassen, zurück zu DOS      | <b>q</b>                         |
| Register   | Register abfragen / Wert zuweisen         | <b>r</b> [Reg]                   |
| Search     | Suche nach Zeichen                        | <b>s</b> Bereich Liste           |
| Trace      | Ausführung und Protokoll der Register     | <b>t</b> [=Adr] [Wert]           |
| Unassemble | Code rückübersetzen in Assembleranw.      | <b>u</b> [Adr] oder u [Bereich]  |
| Write      | Dateien oder abs. Diskettensek. schreiben | <b>w</b> [Adr [Laufw Sek. Sek.]] |

Name in Abhängigkeit des Zustandes des **Flagregisters F** :

| Flagname                         | gesetzt | gelöscht |
|----------------------------------|---------|----------|
| -Overflow (yes / no)             | OV      | NV       |
| -Direction (decrement/increment) | DN      | UP       |
| -Interrupt (enable/disable)      | EI      | DI       |
| -Sign (negative/positive)        | NG      | PL       |
| -Zero (yes / no)                 | ZR      | NZ       |
| -Auxiliary carry (yes / no)      | AC      | NA       |
| -Parity (yes / no)               | PE      | PO       |
| -Carry (yes / no)                | CY      | NC       |

## Softwareentwicklung ( mit Borland TASM - TLINK - TD - TLIB )

Der Assembler-Quelltext **name.ASM** soll mit einem **ASCII-Text-Editor** (Text darf nur ASCII-Zeichen enthalten) erstellt werden ( z.B. DOS-EDIT oder WORD im Nur-Text-Modus )

Der symbolische Assembler übersetzt den Quelltext in den sogn. Objectcode und erzeugt eine Object-Datei **name.OBJ**

Aufruf: **TASM <name> , , ,** falls keine Endung angegeben wird, sucht der Assembler nach name.ASM

und falls eine Listing-Datei **name.LST** mit der Auflistung von Quell-, Maschinencode und Fehlermeldungen gewünscht wird

Aufruf: **TASM /l <name>**

und falls mit dem Turbodebugger auf Quelltextebene debuggt werden soll

Aufruf: **TASM /zi <name>**

Der Binder ( Linker ) hat die Aufgabe, eine oder mehrere Objectdateien zusammenzufügen und in die Betriebssystem-umgebung einzubinden und dabei ist die Erstellung einer Relocationstabelle nötig. In dieser Tabelle sind alle Positionen im Programm enthalten, deren Inhalt zum Zeitpunkt des Ladens durch DOS an die vorgegebene Speicherplatzierung angepaßt werden müssen.

Aufruf: **TLINK <name\_1> + <name\_2> , <name> , Bibliothek.lib** erstellt aus den beiden Objectdateien

und den benötigten Modulen der Bibliothek eine **name\_1.EXE**-Datei

Zusätzlich zur .EXE-datei wird auch noch eine **.MAP-Datei** mit Kreuzreferenzen erzeugt.

Soll eine **.COM- Datei** erzeugt werden, dann kann dies beim Turbo-Linker durch die Option **/t** erreicht werden und es muß nicht wie beim MASM nach dem normalen Linken die Umwandlung nach COM durch das Programm **EXE2BIN** durchgeführt werden (Umwandlung zu einem ladbaren binärem Programm).

Aufruf: **TLINK /t <name>** erzeugt name.COM -Datei

Mit Option **/v** werden sämtliche **symbolischen** Debugger-Informationen bei der Erstellung von **EXE-Dateien** mitgelinkt ( nicht für .COM-Dateien ), damit das Programm mit dem Turbo-Debugger auf der Quellcode-Ebene untersucht werden kann.

Die so entstandenen .EXE oder .COM-Programme können unter DOS durch Eingabe von **name** ausgeführt werden, wobei DOS zuerst nach der name.COM – Datei sucht und dann erst nach der .EXE-Datei.

Für ein komfortables Quelltextdebugging (mit allen Symbolen, im Gegensatz zum DOS-DEBUG )steht der Turbo-Debugger zur Verfügung.

Aufruf: **TD <name>**

Mit **TLIB** lassen sich Bibliotheken aus **.obj-Dateien** bilden, deren Unterprogramme mit **PUBLIC** öffentlich gemacht werden müssen. Im aufrufenden Programm müssen solche externen Programmteile mit **EXTRN** deklariert sein.

**TLIB <name> +datei1.obj + datei2.obj , listdat.lst** bildet mit bildet mit den **obj**-Dateien eine **name.lib** –

Bibliotheksdatei

Beispiel: **TLIB IOLIB +einaus, IOLIB**  
**TLINK hauptprog „, IOLIB**

linkt alle obj.-Dateien der Bibliothek, in denen sich Unterprogramme befinden, die im Hauptprogramm mit EXTRN deklariert wurden, hinzu

## Einführung in die Arbeit mit dem Entwicklungssystem MOPSOS

( funktioniert problemlos nur unter Windows 98 und Windows ME )

für .EXE-Dateien mit Speichermodell SMALL und hinzulinken der Programmbibliothek IOLIB.LIB

Aufruf: D:\MOPSOS> **MOPSOS** <cr>

Funktion mit Taste **F1** bis **F10** aufrufen *oder* mit Cursor <-- oder --> anfahren und mit <cr> ausführen

**F2=Datei** Arbeitsdatei (Name in linker unterer Ecke) einstellen, bleibt bis Neueinstellung mit F2 erhalten

**F3=Edit** Quelltext editieren

für neue Datei: **Datei/Speichern unter** name.ASM Kopie mit F2=Datei zur Arbeitsdatei machen!

**F4=ASM** Assemblieren, bei Fehlermeldung zurück nach Editor

Funktionen können einer Bibliothek iolib.LIB entnommen werden (call name\_x)

INCLUDE iolib.inc baut Deklarationen ein [ EXTRN name\_x:NEAR,.....]

**F5=Link** Linken, bei Fehlermeldung zurück nach Editor

**F6=Go** Programm .EXE starten

zurück nach System mit Makro .EXIT oder Befehl int 20h

Abbruch bei Warten auf Konsole ( INT 21h ): **Strg - C** *oder* **Strg - Break** *oder* **Reset**-Taste

**F8=Debug** Meldung "Program has no symbol table" mit OK ignorieren

CPU-Fenster auf Bildschirmgröße erweitern

- **F7-Trace:** im Einzelschritt jeden Befehl ausführen

- **F8-Step:** im Einzelschritt Unterprogramme übergehen

- nach Ausführung von .STARTUP Menü: View/CPU gibt DS:Datensegment

- **F9-Run:** Programm schnell ausführen

**Neustart:** Menü **Run/Program reset**

32-bit-Register einblenden:

- das Registerfeld anklicken, - lokales Menü: Alt-F10, - Registers 32-bit No mit cr umschalten

Floating Point Register NCP einblenden:

- View / Numeric processor, - NCP-Fenster einblenden / ausblenden

Programm-**Abbruch** in Notfällen:

- **Strg-C** nur, wenn DOS-Eingabefunktion wartet *oder* - **Strg-Break** (2x) *oder* -**Reset**-Taste

Ende des Debuggers mit **File/Quit (Alt-X)**

**F7=List** Übersetzungsliste .LST ausgeben

**F9=DOS** kopieren mit >copy Quelle.ASM, Ziel.ASM Kopie mit F2=Datei zur Arbeitsdatei machen!!!!

zurück nach MOPSOS mit >**EXIT**

**F10=Ende** oder Escape-Taste: zurück nach D:\MOPSOS>

Testprogramm mit **Datei/speichern unter** nach **name.ASM** kopieren und mit **F2**=Datei zur Arbeitsdatei machen

```
;noname.asm Übung Muster für exe.-Programmrahmen mit TASM

escape EQU 1bh ; Endemarke definiert
DOSSEG ; Segmente für DOS
.MODEL small ; 1 Datensegment 1 Codesegment .EXE-Programm
.486 ; voller Befehlssatz ab 486
.STACK 256 ; Stapel der Größe 256 Bytes
.DATA ; Datensegment
prompt DB '>' ; Eingabeprompt
DB ' Hier liegen die Testdaten im Datensegment ',0
.CODE ; Codesegment
.STARTUP ; Makro: Segmentregister anlegen
mov dl,prompt ; DL <- Eingabeprompt
mov ah,2 ; Funktion Konsolaausgabe
int 21h ; DOS
schleife: mov ah,1 ; Funktion warten auf Konsole mit Echo
int 21h ; DOS AL <= Zeichen von Konsole
cmp al,escape ; AL == Endemarke Escape-Taste ?
je ausgang ; ja: fertig
jmp schleife ; nein: Leseschleife bis Endemarke

ausgang: .EXIT ; Makro: zurück nach DOS

; hier müssen die benutzerdefinierten internen Unterprogramme liegen

END ; Ende des Quelltextes
```

### Unterprogrammbibliotheken IOLIB.LIB bzw. EINAUS.LIB

```
; iolib.inc und einaus.inc enthalten die EXTRN Definitionen
; Zuordnung mit INCLUDE iolib.inc bzw. einaus.inc Aufruf mit CALL name
EXTRN putch:NEAR ; AL -> Konsole ausgeben
EXTRN putblank:NEAR ; lz ausgeben
EXTRN putprompt:NEAR ; cr lf AL -> Konsole ausgeben
EXTRN getch:NEAR ; AL <- Konsole mit warten ohne Echo
EXTRN getche:NEAR ; AL <- Konsole mit warten und Echo
EXTRN puts:NEAR ; DS:AX = Stringadresse bis 0 ausgeben
EXTRN gets:NEAR ; DS:AX = Stringadresse bis cr lesen
EXTRN kbhit:NEAR ; AL == 0: kein Zeichen AL == -1 abholen
EXTRN balaus:NEAR ; AL -> 8 Binärziffern ausgeben
EXTRN haxein:NEAR ; AX <= 4 Hexaziffern lesen
EXTRN halaus:NEAR ; AL -> 2 Hexaziffern ausgeben
EXTRN ausnib:NEAR ; AL -> linkes Nibble ASCII codiert ausg.
EXTRN haxaus:NEAR ; AX -> 4 Hexaziffern ausgeben
EXTRN daxein:NEAR ; AX <- Dezimalzahl < 65536 lesen
EXTRN daxaus:NEAR ; AX -> Dezimalzahl ausgeben
EXTRN dedez:NEAR ; AL -> ASCII decodieren C=1: Nicht-Dezi
EXTRN d32aus:NEAR ; DX:AX -> dezimal ohne führende Nullen
EXTRN zufall:NEAR ; AX=alte -> AX=neue BX=bereich -> BX=<bereich
EXTRN wuerfel:NEAR ; AX <= 40h:6ch Timer-Tic-Variable
EXTRN taste:NEAR ; AL == 0: kein Zeichen wie kbhit
EXTRN grafinit:NEAR ; Grafikbildschirm ohne Parameter
EXTRN mausinit:NEAR ; Maus initialisieren ohne Parameter
EXTRN lclick:NEAR ; AX = X-Pos BX = Y-Pos linke Maustaste
EXTRN mclose:NEAR ; Maus schließen ohne Parameter
```

## Beispielprogramm für Bibliotheksfunktionen

```
; mopueb.asm  MOP-Übung externe Uprogramme in iolib.LIB (einaus.LIB)
                INCLUDE einaus.inc; Deklarationen (extrn) für iolib.LIB
                DOSSEG           ; Segmente für DOS
                .MODEL  small     ; 1 Datensegment 1 Codesegment
                .486             ; voller Befehlssatz ab 486
                .STACK  256       ; Stapelsegment 256 Bytes
                .DATA            ; Datensegment
text            DB  10,13,"Eingabe bis Esc > ",0
ende           DB  10,13,"Auf Wiedersehen > ",0
                .CODE            ; Codesegment
                .STARTUP         ; Makro: Segmentregister anlegen
schlei:        lea    ax,text     ; AX <= String-Anfangs-Adresse
                call   puts       ; Meldung ausgeben
                call   getch      ; AL <= Zeichen lesen ohne Echo
                cmp    al,1bh     ; Ende mit Zeichen Esc ?
                je     mensa      ; ja: Ende der Leseschleife
                cmp    al,'a'     ; nein: untere Grenze ?
                jb     aus        ; kleiner: ausgeben
                cmp    al,'z'     ; obere Grenze ?
                ja     aus        ; grösser: ausgeben
                and    al,1101111b ; maskieren klein -> gross
aus:           call   putch       ; AL Zeichen ausgeben
                call   putblank   ; Leerzeichen
                call   balaus     ; AL binär ausgeben
                call   putblank   ; Leerzeichen
                call   halaus     ; AL hexa ausgeben
                call   putblank   ; Leerzeichen
                xor     ah,ah      ; AH = 0
                call   daxaus     ; AX dezimal ausgeben
                jmp    schlei     ; und weiter bis Esc
mensa:        lea    ax,ende     ; AX <= String-Anfangs-Adresse
                call   puts       ; Text ausgeben
                .EXIT           ; Makro: zurück nach DOS
                END             ; Ende des Quelltextes
```

**Bibliothek** einaus.LIB = iolib.LIB Deklarationen in einaus.INC

Zuordnung der EXTRN Deklarationen im Quellfile: INCLUDE einaus.INC

| Bezeichner | Funktion                        | Eingabe            | Rückgabe         |
|------------|---------------------------------|--------------------|------------------|
| getch      | warten Zeichen ohne Echo lesen  | -                  | AL = Zeichen     |
| getche     | warten Zeichen mit Echo lesen   | -                  | AL = Zeichen     |
| putch      | Zeichen auf Konsole ausgeben    | AL = Zeichen       | -                |
| putblank   | Leerzeichen (Blank) ausgeben    | -                  | -                |
| gets       | String bis Steuerz. < 20h lesen | AX = Pufferadresse | AL = Endezeichen |
| puts       | String bis Endemarke Null ausg. | AX = Pufferadresse | -                |
| halaus     | AL in 2 Hexaziffern ausgeben    | AL = Byte          | -                |
| balaus     | AL in 8 Binärziffern ausgeben   | AL = Byte          | -                |
| daxaus     | 16 bit in AX dezimal ausgeben   | AX = Wert          | -                |

## Assembler

**Operatoren** werden vom Assembler zur Übersetzungszeit ausgewertet und liefern Werte.

**Direktiven** ( Assembleranweisungen ) erzeugen keinen Code, sondern steuern die Übersetzung.

| Operator / Direktive       | Anwendung                                                                    | Beispiel:                                                |
|----------------------------|------------------------------------------------------------------------------|----------------------------------------------------------|
| + - * /                    | Vor- und Rechenzeichen                                                       | mov ax, tab + 2                                          |
| :                          | Segmentvorsatz CS: DS: ES: SS:                                               | mov ax, CS:wert                                          |
| [ ]                        | kennzeichnet Adressierungsart                                                | mov ax, [bx] ; indirekt<br>mov ax, DS:[0000] ; direkt    |
| OFFSET                     | liefert Abstand im Segment                                                   | mov bx, OFFSET tab                                       |
| SEG ausdruck               | liefert die Segmentadresse eines Ausdrucks                                   | mov ax, SEG tab                                          |
| BYTE PTR                   | Datentyp Byte                                                                | inc BYTE PTR [bx]                                        |
| WORD PTR                   | Datentyp Wort                                                                | inc WORD PTR [si]                                        |
| NEAR PTR                   | Sprung oder Aufruf innerhalb des Segments                                    | call NEAR PTR putch                                      |
| FAR PTR                    | Sprung / Aufruf außerhalb des Segments (interseg)                            | call FAR PTR getch                                       |
| WORD PTR                   | Sprung indirekt mit Wortadresse ( intrasegment)                              | jmp WORD PTR [di]                                        |
| anzahl DUP (wert)          | Wiederholungsfaktor für Datenvereinbarung                                    | tab DB 10 DUP ( '*' )                                    |
| ?                          | nicht vorbesetzte Variable                                                   | tab DB 10 DUP ( ? )                                      |
| \$                         | laufender Adreßzähler des Assemblers                                         | jmp \$ + 10 ; besser<br>Sprungziel                       |
| Bezeichner EQU wert        | vereinbart Bezeichner für Ausdruck                                           | escape EQU 1B h                                          |
| ORG ausdruck               | legt aktuelle Position im aktuellen Segment fest                             | ORG 100 h ; Startadresse .COM                            |
| END [startadresse]         | markiert Ende des Quelltextes                                                | END ; ohne Startadresse<br>END anfang ; mit Startadresse |
| . DOSSEG<br>DOSSEG         | MASM: Anordnung der Segmente für DOS<br>TASM: Anordnung der Segmente für DOS | . DOSSEG<br>DOSSEG                                       |
| . MODEL typ                | tiny small medium compact large huge                                         | . MODEL small                                            |
| . cpu                      | Register und Befehle für 286 386 486                                         | . 486                                                    |
| . STACK länge              | Grösse des Stapels in Byte                                                   | . STACK 256                                              |
| . DATA                     | es folgt das Datensegment                                                    | . DATA                                                   |
| DB                         | Speicher für ein Byte ( 8 Bit )                                              | maxi DB 55 h ; vorbesetzt<br>DB ? ; undefiniert          |
| DW                         | Speicher für ein Wort ( 16 Bit )                                             | mini DW 1234 h ; Konstante<br>zeig DW maxi ; Adresse     |
| DD                         | Speicher für ein Doppelwort ( 32 Bit )                                       | x DD 100000 ; ganz                                       |
| DQ                         | Speicher für ein Vierfachwort ( 64 Bit )                                     | y DQ 47.11 ; reell                                       |
| DT                         | Speicher für ein Zehnerwort ( 80 Bit)                                        | z DT 1E100 ; reell                                       |
| . CODE                     | es folgt das Codesegment                                                     | . CODE                                                   |
| . STARTUP                  | Systemmakro lädt DS SS SP                                                    | . STARTUP                                                |
| . EXIT                     | Systemmakro zurück nach DOS                                                  | : EXIT                                                   |
| SEGMENT [attribut]<br>ENDS | Rahmen für Segment                                                           | daten SEGMENT<br>daten ENDS                              |
| PROC [attribut]<br>ENDP    | Rahmen für Prozedur ( Unterprogramm )                                        | putch PROC<br>putch ENDP                                 |
| MACRO [parameter]<br>ENDM  | Rahmen für Makro                                                             | aus MACRO<br>ENDM                                        |
| LOCAL bezeichner           | lokaler Bezeichner in PROC bzw. MACRO                                        | LOCAL schleife                                           |
| PUBLIC: typ liste          | Bezeichner auch in anderen Moduln verfügbar                                  | PUBLIC otto : NEAR                                       |
| EXTRN : typ liste          | Bezeichner liegt in anderem Modul                                            | EXTRN otto : NEAR, max :<br>NEAR                         |
| INCLUDE                    | Textdatei in Quelltext einfügen                                              | INCLUDE einaus.inc                                       |

## Übersicht der 8086-Assemblerbefehle ( Maschinenbefehle )

| Befehl     | Funktion                                         |
|------------|--------------------------------------------------|
| AAA        | ASCII adjust AL for addition                     |
| AAD        | ASCII adjust for division                        |
| AAM        | ASCII adjust for multiply                        |
| AAS        | ASCII adjust for subtraction                     |
| ADC X1,X2  | Add with carry                                   |
| ADD X1,X2  | Addition                                         |
| AND X1,X2  | Logical-AND                                      |
| CALL NAER  | CALL im Segment                                  |
| CALL FAR   | CALL über Segmentgrenzen                         |
| CBW        | Convert byte in word                             |
| CLC        | Clear carry- flag                                |
| CLD        | Clear direction flag                             |
| CLI        | Clear interrupt enable flag                      |
| CMC        | Complement carry flag                            |
| CMP X1,X2  | Compare                                          |
| CMPS X1,X2 | Compare Strings                                  |
| CWD        | Convert word to doubleword                       |
| DAA        | Decimal adjust AL after addition                 |
| DAS        | Decimal adjust AL after subtraction              |
| DEC X1     | Decrement                                        |
| DIV X1     | Unsigned divide                                  |
| HLT        | Halt                                             |
| IDIV X1    | Signed divide                                    |
| IMUL X1    | Signed multiply                                  |
| IN X1,X2   | Input byte/word from port                        |
| INC X1     | Increment X1 um 1                                |
| INT 3      | Interrupt 3                                      |
| INT xx     | Interrupt xx                                     |
| INTO       | Interrupt on overflow                            |
| IRET       | Interrupt return                                 |
| JA cb      | Jump short if above (CF=0 and ZF=0)              |
| JAE cb     | Jump short if above or equal (CF=0)              |
| JB cb      | Jump short if below (CF=1)                       |
| JBE cb     | Jump short if below or equal (CF=1 or ZF=1)      |
| JC cb      | Jump short if carry (CF=1)                       |
| JCXZ cb    | Jump short if CX register is zero                |
| JE cb      | Jump short if equal (ZF=1)                       |
| JG cb      | Jump short if greater (ZF=0 and SF=OF)           |
| JGE cb     | Jump short if greater or equal (SF=OF)           |
| JL cb      | Jump short if less (SF/=OF)                      |
| JLE cb     | Jump short if less or equal (ZF=1 or SF/=OF)     |
| JMP cb     | Jump short                                       |
| JMP FAR    | Jump far (4-byte address)                        |
| JMP NEAR   | Jump near                                        |
| JNA cb     | Jump short if not above (CF=1 or ZF=1)           |
| JNAE cb    | Jump short if not above or equal (CF=1)          |
| JNB cb     | Jump short if not below (CF=0)                   |
| JNBE cb    | Jump short if not below or equal (CF=0 and ZF=0) |
| JNC cb     | Jump short if not carry (CF=0)                   |
| JNE cb     | Jump short if not equal (ZF=0)                   |
| JNG cb     | Jump short if not greater (ZF=1 or SF/=OF)       |
| JNGE cb    | Jump short if not greater or equal (SF/=OF)      |
| JNL cb     | Jump short if not less (SF=OF)                   |
| JNLE cb    | Jump short if not less or equal (ZF=0 and SF=OF) |
| JNO cb     | Jump short if not overflow (OF=0)                |
| JNP cb     | Jump short if not parity (PF=0)                  |
| JNS cb     | Jump short if not sign (SF=0)                    |
| JNZ cb     | Jump short if not zero (ZF=0)                    |
| JO cb      | Jump short if overflow (OF=1)                    |
| JP cb      | Jump short if parity (PF=1)                      |
| JPE cb     | Jump short if parity even (PF=1)                 |

---

|                |                                                          |
|----------------|----------------------------------------------------------|
| JPO cb         | Jump short if parity odd (PF=0)                          |
| JS cb          | Jump short if sign (SF=1)                                |
| JZ cb          | Jump short if zero (ZF=1)                                |
| LAHF           | Load: AH= flags                                          |
| LDS X1 ,X2     | Load pointer using DS                                    |
| LEA X1 ,X2     | Load effective adress                                    |
| LES X1 ,X2     | Load pointer using ES                                    |
| LODS           | Load stringbyte                                          |
| LODSW          | Load stringword                                          |
| LOOP XX        | Loop DEC CX; jump short if CX/=0                         |
| LOOPE XX       | Loop Equal DEC CX; jump short if CX/=0 and equal (ZF=1)  |
| LOOPNE XX      | Loop Not Equal DEC CX; jump short if CX/=0 and not equal |
| LOOPNZ XX      | Loop Not Zero DEC CX; jump short if CX/=0 and ZF=0       |
| LOOPZ XX       | Loop Zero DEC CX; jump short if CX/=0 and zero (ZF=1)    |
| MOV X1 ,X2     | Move                                                     |
| MOVSB          | Move Stringbyte                                          |
| MOVSW          | Move Stringword                                          |
| MUL            | Multiply unsigned                                        |
| NEG            | Negate                                                   |
| NOP            | No Operation                                             |
| NOT            | NOT-Vergleich                                            |
| OR X1 ,X2      | Logical-OR                                               |
| OUT X1 ,X2     | Output to port                                           |
| POP X1         | POP Register from stack                                  |
| POPF           | Pop Flags                                                |
| PUSH X1        | PUSH Register to stack                                   |
| PUSHF          | PUSH Flags                                               |
| RCL xx,l       | Rotate left carry                                        |
| RCR xx,l       | Rotate right carry                                       |
| REP (prefix)   | Repeat                                                   |
| REPE (prefix)  | Repeat equal                                             |
| REPNE (prefix) | Repeat not equal                                         |
| REPNZ (prefix) | Repeat not zero                                          |
| REPZ (prefix)  | Repeat zero                                              |
| RETF           | Return far                                               |
| RET            | Return near                                              |
| ROL xx,l       | Rotate left                                              |
| ROR xx,l       | Rotate right                                             |
| SAHF           | Store AH into flags                                      |
| SAL xx,l       | Shift arithmetik left                                    |
| SAR xx,l       | Shift arithmetik right                                   |
| SBB X1 ,X2     | Subtract with borrow                                     |
| SCAS           | Compare Strings                                          |
| SHL xx,l       | Shift left                                               |
| SHR xx,l       | Shift right                                              |
| STC            | Set carry flag                                           |
| STD            | Set direction flag                                       |
| STI            | Set interrupt enable flag                                |
| STOS xx        | Store String                                             |
| SUB X1 ,X2     | Subtract                                                 |
| TEST X1 ,X2    | Test                                                     |
| XCHG X1 ,X2    | Exchange                                                 |
| XLAT xx        | Translate                                                |
| XOR X1 ,X2     | Exclusive-OR                                             |

## Software-Interrupts

| Interrupt | Beschreibung |
|-----------|--------------|
|-----------|--------------|

|     |                                                                          |
|-----|--------------------------------------------------------------------------|
| 00h | DIVIDE ERROR (Division durch 0)                                          |
| 01h | SINGLE STEP; (80386+), DEBUGGING                                         |
| 02h | NON-MASKABLE INTERRUPT                                                   |
| 03h | BREAKPOINT (CPU)                                                         |
| 04h | INTO DETECTED OVERFLOW (CPU-Obenauf)                                     |
| 05h | PRINTSCREEN                                                              |
| 06h | INVALID OPCODE (80286+)                                                  |
| 07h | PROCESSOR EXTENSION, reserviert (80286+)                                 |
| 08h | IRQ0, SYSTEM TIMER (80286+)                                              |
| 09h | IRQ1, KEYBOARD DATA READY (80286+)                                       |
| 0Ah | IRQ2, LPT2/EGA,VGA, IRQ9 (80286+)                                        |
| 0Bh | IRQ3, COM2 (80286+)                                                      |
| 0Ch | IRQ4, COM1 (80286+)                                                      |
| 0Dh | IRQ5, HARDDISK, LPT2 (80286+)                                            |
| 0Eh | IRQ6, DISC CONTROLLER (80286+)                                           |
| 0Fh | IRQ7, PARALLEL PRINTER                                                   |
| 10h | GRAPHIC (80286+)                                                         |
| 11h | BIOS, GET EQUIPMENT LIST (80486+)                                        |
| 12h | BIOS, GET MEMORY SIZE                                                    |
| 13h | LAUFWERKSFUNKTIONEN EIN/AUSGABE                                          |
| 14h | SERIELLE SCHNITTSTELLEN                                                  |
| 15h | CASSETTEN RECORDER (8088), erweiterte Aufrufe                            |
| 16h | KEYBOARD EIN/AUSGABE                                                     |
| 17h | PRINTER EIN/AUSGABE                                                      |
| 18h | DISKLESS BOOT HOOK (START CASSETTE BASIC), diverse Funktionen, ROM-BASIC |
| 19h | SYSTEM, BOOTSTRAP LOADER                                                 |
| 1Ah | TIMER EIN/AUSGABE, PCI-I/O                                               |
| 1Bh | KEYBOARD, CONTROL-BREAK HANDLER                                          |
| 1Ch | TIMER, SYSTEM TIMER TICK                                                 |
| 1Dh | SYSTEM DATA, GRAPHIC PARAMETERS                                          |
| 1Eh | SYSTEM DATA, DISKETTE PARAMETERS                                         |
| 1Fh | SYSTEM DATA, 8x8 GRAPHICS FONT                                           |

## DOS

|           |                                          |
|-----------|------------------------------------------|
| 20h       | DOS, TERMINATE PROGRAM                   |
| 21h       | DOS, FUNCTION CALLS                      |
| 22h       | DOS, PROGRAM TERMINATION ADDRESS         |
| 23h       | DOS, CONTROL-C/CONTROL-BREAK HANDLER     |
| 24h       | DOS, CRITICAL ERROR HANDLER              |
| 25h       | DOS, ABSOLUTE DISK READ                  |
| 26h       | DOS, ABSOLUTE DISK WRITE                 |
| 27h       | DOS, TERMINATE AND STAY RESIDENT         |
| 28h       | DOS, DOS IDLE INTERRUPT                  |
| 29h       | DOS, FAST CONSOLE OUTPUT                 |
| 2Ah       | DOS, CRITICAL ERROR, NETBIOS             |
| 2Bh - 2Dh | DOS, RESERVED                            |
| 2Eh       | DOS, PASS COMMAND TO COMMAND INTERPRETER |
| 2Fh       | DOS, MULTIPLEXER                         |
| 30h       | DOS, FAR JMP INSTRUCTION                 |
| 33h       | MOUSE                                    |
| 34h-3Fh   | FLOATING POINT EMULATION                 |

**Interrupt      Beschreibung**

**Anwender**

|         |                                                                        |
|---------|------------------------------------------------------------------------|
| 40h     | DISKETTENFUNKTIONEN, umgeleiteter Interrupt 13                         |
| 41h     | SYSTEM DATA, HARD DISK 0 PARAMETER TABLE                               |
| 42h     | GRAPHIC SERVICES (EGA,VGA), umgeleiteter Interrupt 10                  |
| 43h     | GRAPHIC DATA (EGA,MCGA,VGA)                                            |
| 44h     | GRAPHIC EGA-CHARACTERS, NOVELL NETWARE                                 |
| 46h     | SYSTEM DATA, HARD DISK 1 DRIVE PARAMETER TABLE                         |
| 47h-4Eh | Diverse Funktionen                                                     |
| 4Fh     | SCSI, Common Access Method                                             |
| 50h     | Umgeleiteter IRQ0                                                      |
| 51h     | Umgeleiteter IRQ1                                                      |
| 52h     | Umgeleiteter IRQ2                                                      |
| 53h     | Umgeleiteter IRQ3                                                      |
| 54h     | Umgeleiteter IRQ4                                                      |
| 55h     | Umgeleiteter IRQ5                                                      |
| 56h     | Umgeleiteter IRQ6                                                      |
| 57h     | Umgeleiteter IRQ7                                                      |
| 58h     | Umgeleiteter IRQ8/0                                                    |
| 59h     | Umgeleiteter IRQ9/1                                                    |
| 5Ah     | Umgeleiteter IRQ10/2                                                   |
| 5Bh     | Umgeleiteter IRQ11/3                                                   |
| 5Ch     | Umgeleiteter IRQ12/4                                                   |
| 5Dh     | Umgeleiteter IRQ13/5                                                   |
| 5Eh     | Umgeleiteter IRQ14/6                                                   |
| 5Fh     | Diverse Funktionen                                                     |
| 60h-66h | Reserviert für spezielle Interrupt-Verarbeitung und diverse Funktionen |
| 67h     | LIM EMS, diverse Funktionen                                            |
| 68h-6Ch | Diverse Funktionen                                                     |
| 6Dh     | On Board VGA                                                           |
| 6Eh     | Netzwerk API                                                           |
| 6Fh     | Novell NetWare                                                         |
| 70h     | IRQ8, CMOS REAL-TIME CLOCK                                             |
| 71h     | IRQ9, umgeleitet vom BIOS zu INT0A                                     |
| 72h     | IRQ10, reserviert                                                      |
| 73h     | IRQ11, reserviert                                                      |
| 74h     | IRQ12, diverse Funktionen                                              |
| 75h     | IRQ13, MATH COPROCESSOR EXCEPTION (286+)                               |
| 76h     | IRQ14, HARD DISK CONTROLLER (286+)                                     |
| 77h     | IRQ15, diverse Funktionen                                              |
| 78h     | DOS Extender; diverse Funktionen                                       |
| 79h     | Diverse Funktionen                                                     |
| 7Ah     | Novell NetWare, diverse Funktionen                                     |
| 7Bh-7Fh | Diverse Funktionen                                                     |
| 80h-F0h | Reserviert für IBM-Basic; diverse Funktionen                           |
| F1h-FFh | Reserviert für User-Interrupt                                          |

## Interrupt - Programmierung mit 80x86

**Beispiel:** Timer 0 – IRQ0 (System-Uhr Tic im PC ) inkrementiert BCD-Zähler (Byte) bei jedem durch Timer 0 bewirkten Einsprung in die zugehörige ISR (**Interruptcontroller ist im PC bereits initialisiert** ).

**Interruptvektor setzen** für jede aktive Interruptanforderung

```
push  es
mov   ax,0
mov   es,ax
mov   es : 20 h , offset TIMER0 ; ISR           poke (0x00, 0x20, FP_OFF( TIMER0) + 0x100 ) ;
mov   es : 22 h, cs                    poke (0x00, 0x22, FP_SEG( TIMER0) );
pop   es
```

Im PC kann das Setzen des Interrupt-Vektors einfacher durch die **DOS-Funktion 25h INT 21h** erfolgen

```
mov  ah, 25h
mov  al, 08          ; Beispiel einer Interrupt –Vektor-Nummer
mov  dx, offset TIMER0 ; DS:DX FAR-Zeiger auf ISR -> mov  cx, cs ; mov  ds, cx
int  21h             ; da für .COM-Programme DS = CS
```

für den Fall, daß ein existierender Interrupt-Vektor auf eine neue ISR verbogen werden und nach Abarbeitung der neuen ISR zur alten ISR verzweigt werden oder vor Beendigung des Programms der alte Vektor wiederhergestellt werden soll, kann der alte Interrupt-Vektor mittels der **DOS-Funktion 35h INT 21h** gelesen und gerettet werden.

```
mov  ax, 3508h
int  21h
mov  INTR08_old, bx ;gelesenen Intr-Vektor speichern
mov  INTR08_old + 2, es
```

Beispiel einer Wiederherstellung vor Programmende

```
mov  dx, INTR08_old
mov  ax, INTR08_old + 2
mov  ds, ax
mov  ax, 2508h
int  21h
```

**selektive Freigabe** der aktiven Interruptquelle im PIC 8259 (bzw bei Beendigung des Programms **selektiv sperren**)

```
in   al, 21 h          ; Maskenregister           outp ( 0x21, ( inp ( 0x21 ) & 0xfd ) );
and  al, 11111110 b    ; IMR.0 löschen gibt IRQ0 frei
out  21 h, al
```

**generelle Freigabe** des INTR-Eingangs des 80x86 durch den Befehl **STI** -> I-Flag = 1 **enable ()** ;  
( bzw. vor Programmbeendigung die **generelle Sperre** durch **CLI** -> I-Flag = 0 ) **disable ()** ;  
generelle Freigabe des INTR-Eingangs ist im PC-Betrieb grundsätzlich gegeben ( Timer, Tastatur, Disk usw. )

**Interrupt-Service-Routine ISR** für jeden aktiven Interrupt:

- diese ISR muß **alle Register !!!!** unverändert zurückliefern
- falls Unterbrechung dieser ISR möglich sein soll, generelle Freigabe des INTR-Eingang mit STI-Befehl
- vor Verlassen der ISR den Interrupt mit EOI-Befehl bestätigen
- und die ISR mit einem IRET-Befehl beenden

```
TIMER0 PROC
    push    ax
    ( sti    ; damit unterbrechbar )
    mov     al, CS:Zaehler
    inc     al
    daa
    out     CS:Zaehler,al
    mov     al, 20 h ;
    out     20 h, al ; unspezifischer EOI fuer Master
EOI
    pop     ax
    iret    ; return from Interrupt
TIMER0 ENDP
Zaehler    DB    0

void interrupt TIMER0(void); // Prototyp
void main (void) // Hauptfunktion
{ ----- }
void interrupt TIMER0 (void )
{
    asm
    { in     al, APORT // Beispiel fuer
      inc    al // Port-Ausgabe
      daa    // beim MVUS 8088
      out    APORT,al }
    outp ( 0x20, 0x60 ); // spezifischer
}
```

## BIOS-Funktionen

Über die Interrupts 10h bis 1Ah können die verschiedenen Funktionen erreicht werden, die das ROM-BIOS zur grundlegenden Kommunikation zwischen einem Programm und der Hardware zur Verfügung stellt.

Es ist zu beachten, daß die verschiedenen Funktionen des Interrupts 13h getrennt nach ihrem Einsatz in Bezug auf Disketten- und Festplattenlaufwerke in zwei verschiedenen Abschnitten aufgeführt werden.

## Übersicht der BIOS-Funktionen

### Interrupt 10h      Bildschirm

|         |                                               |
|---------|-----------------------------------------------|
| 00h     | Setzen des Video-Modus                        |
| 01h     | Definition des Erscheinungsbildes des Cursors |
| 02h     | Positionierung des Cursors                    |
| 03h     | Auslesen der Cursor-Position                  |
| 04h     | Auslesen der Lichtstiftposition               |
| 05h     | Auswahl der aktuellen Bildschirmseite         |
| 06h     | Textzeilen nach oben schieben (scrollen)      |
| 07h     | Textzeilen nach unten schieben (scrollen)     |
| 08h     | Auslesen eines Zeichens/Farbe                 |
| 09h     | Schreiben eines Zeichens/Farbe                |
| 0Ah     | Schreiben eines Zeichens                      |
| 0Bh/00h | Auswahl der Rahmen-/Hintergrundfarbe          |
| 0Bh/01h | Auswahl der Farbpalette                       |
| 0Ch     | Schreibe Grafikpunkt                          |
| 0Dh     | Lese Grafikpunkt                              |
| 0Eh     | Schreiben eines Zeichens                      |
| 0Fh     | Auslesen des Video-Modus                      |
| 13h     | Ausgabe einer Zeichenkette                    |

### Interrupt 11h      Feststellung der Konfiguration

### Interrupt 12h      Feststellung der Speichergröße

### Interrupt 13h      Diskette

|     |                                      |
|-----|--------------------------------------|
| 00h | Reset                                |
| 01h | Status Lesen                         |
| 02h | Lesen                                |
| 03h | Schreiben                            |
| 04h | Verifizieren                         |
| 05h | Formatieren                          |
| 06h | Festplatte                           |
| 07h | Festplatte                           |
| 08h | Format abfragen                      |
| 09h | Festplatte                           |
| 0Ah | Festplatte                           |
| 0Bh | Festplatte                           |
| 0Ch | Festplatte                           |
| 0Dh | Festplatte                           |
| 0Eh | Festplatte                           |
| 0Fh | Festplatte                           |
| 10h | Festplatte                           |
| 11h | Festplatte                           |
| 12h | Festplatte                           |
| 13h | Festplatte                           |
| 14h | Festplatte                           |
| 15h | Feststellung des Laufwerktyps        |
| 16h | Feststellung eines Diskettenwechsels |
| 17h | Diskettenformat festlegen            |
| 18h | Diskettenformat festlegen            |

**Interrupt 13h      Festplatte**

|     |                               |        |
|-----|-------------------------------|--------|
| 00h | Reset                         |        |
| 01h | Status lesen                  |        |
| 02h | Lesen                         |        |
| 03h | Schreiben                     |        |
| 04h | Verifizieren                  |        |
| 05h | Formatieren                   |        |
| 08h | Format erfragen               |        |
| 09h | Anpassung fremder Laufwerke   |        |
| 0Ah | Erweitertes Lesen             |        |
| 0Bh | Erweitertes Schreiben         |        |
| 0Ch | Schreib-/Lesekopf bewegen     |        |
| 0Dh | Reset                         |        |
| 0Eh | Controller-Lese-Test          | (PS/2) |
| 0Fh | Controller-Schreib-Test       | (PS/2) |
| 10h | Ist das Laufwerk bereit?      |        |
| 11h | Rekalibrieren des Laufwerks   |        |
| 12h | Controller-RAM-Test           | (PS/2) |
| 13h | Laufwerk-Test                 | (PS/2) |
| 14h | Controller-Diagnose           |        |
| 15h | Feststellung des Laufwerktyps |        |

**Interrupt 14h      Serielle Schnittstelle**

|     |                  |
|-----|------------------|
| 00h | Initialisierung  |
| 01h | Zeichen ausgeben |
| 02h | Zeichen einlesen |
| 03h | Status erfragen  |

**Interrupt 15h      Alter Kassetten-Interrupt**

|         |                                                  |
|---------|--------------------------------------------------|
| 83h     | Flag nach Zeitintervall setzen                   |
| 84h/00h | Abfrage des Status der Feuerknöpfe der Joysticks |
| 84h/01h | Abfrage der Stellung der Joysticks               |
| 85h     | SysReq-Taste betätigt                            |
| 86h     | Warten                                           |
| 87h     | Speicherbereiche verschieben                     |
| 88h     | Speichergröße über 1 MByte ermitteln             |
| 89h     | Umschaltung in den Protected Mode                |

**Interrupt 16h      Tastatur**

|     |                                           |
|-----|-------------------------------------------|
| 00h | Zeichen auslesen                          |
| 01h | Zeichen vorhanden?                        |
| 02h | Status der Tastatur erfragen              |
| 03h | Wiederholrate einstellen                  |
| 05h | Tastendruck simulieren                    |
| 10h | Tastaturabfrage für erweiterte Tastaturen |
| 11h | Tastaturabfrage für erweiterte Tastaturen |

**Interrupt 17h      (paralleler) Drucker**

|     |                              |
|-----|------------------------------|
| 00h | Zeichen ausgeben             |
| 01h | Drucker initialisieren       |
| 02h | Status des Druckers erfragen |

**Interrupt 18h      ROM-BASIC**

**Interrupt 19h      Booten des Rechners**

**Interrupt 1Ah      Datum und Zeit**

|     |                                          |
|-----|------------------------------------------|
| 00h | Zeit-Zähler auslesen                     |
| 01h | Zeit-Zähler setzen                       |
| 02h | Auslesen der Echtzeit-Uhr                |
| 03h | Setzen der Echtzeit-Uhr                  |
| 04h | Auslesen des Datums aus der Echtzeit-Uhr |
| 05h | Setzen des Datums der Echtzeit-Uhr       |
| 06h | Alarmzeit setzen                         |
| 07h | Alarmzeit löschen                        |

## BIOS-Interrupt-Routinen für die Tastatur Aufruf: `int 16h`

| Funktion                                                           | Eingabe              | Rückgabe                                                                                                                    |
|--------------------------------------------------------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------|
| AH = 0 Zeichen lesen                                               | -                    | wartet auf Taste ohne Echo<br>AH = Scan-Code AL = ASCII-Code                                                                |
| AH = 1 Tastatur testen<br>ohne warten !!                           | -                    | Ergebnis == 0 (Z = 1): kein Zeichen im Puffer<br>Ergebnis != 0 (Z = 0): Zeichen im Puffer und AX<br>abholen mit Fktn AH = 0 |
| AH = 2 TastaturStatus lesen                                        | -                    | AL = Status der Modifizierertasten 40:17 h<br>Bit AL.x = 1 -> Taste betätigt bzw Modus ein                                  |
| AH = 3 Wiederholungsrate<br>und Verzögerung einstellen             | AL=5<br>BL =<br>BH = | Wiederholungsrate: 00h=30 Wps bis 1Fh = 2,0 Wps<br>Verzögerung : 00h=250ms ; 01h=500ms ; 11h = 1 Sek                        |
| AH = 5 Tastendruck simulieren<br>durch Schreiben in Tastaturpuffer | CH =<br>CL =         | Scancode in den Tastaturpuffer schreiben<br>ASCII-Code AL=0 ok; AL = 1 -> Tastaturpuffer voll                               |
| AH = 10h Zeichen lesen<br>von erweiterter Tastatur                 | -                    | AH = Scan-Code AL = ASCII-Code<br>entspricht Fktn. 00, unterstützt erweiterte Tastatur                                      |
| AH = 11h Pufferstatus<br>für erweiterte Tastatur                   | -                    | Ergebnis == 0 (Z = 1): kein Zeichen im Puffer<br>Ergebnis != 0 (Z = 0): Zeichen im Puffer und AX<br>entspricht Funktion 01  |
| AH = 12h TastaturStatus lesen<br>der erweiterten Tastatur          | -                    | AL = 1. Umschaltstatusbyte 40:17 h<br>AH = 2. Umschaltstatusbyte 40:18 h                                                    |

Aufbau des Statusbytes ( 40 : 17 h) der **Modifizierertasten**:

<----- Zustandstasten(Modus) -----> <-----nur mit anderen Tasten zusammen----->

|       |          |         |            |     |      |            |             |
|-------|----------|---------|------------|-----|------|------------|-------------|
| Eingf | CapsLock | NumLock | ScrollLock | Alt | Strg | Shft links | Shft rechts |
|-------|----------|---------|------------|-----|------|------------|-------------|

Aufbau des 2. Statusbytes ( 40 : 18 h) der erweiterten Tastatur

<----- Tasten gedrückt = 1 ----->

|         |          |         |        |       |        |           |            |
|---------|----------|---------|--------|-------|--------|-----------|------------|
| Einfüge | CapsLock | NumLock | Scroll | Pause | Sysreq | Alt links | Strg links |
|---------|----------|---------|--------|-------|--------|-----------|------------|

### 40:96 MF2-Tastatur (Tastaturstatusbyte)

|                           |                             |                              |                       |                  |                   |                           |                           |
|---------------------------|-----------------------------|------------------------------|-----------------------|------------------|-------------------|---------------------------|---------------------------|
| Tastatur-ID Abfrage läuft | erstes ID-Zeichen empfangen | Num-Lock-Modus wenn ID = MF2 | MF2-Tastatur gefunden | rechte Alt-Taste | rechte Ctrl Taste | Code E0 zuletzt empfangen | Code E1 zuletzt empfangen |
|---------------------------|-----------------------------|------------------------------|-----------------------|------------------|-------------------|---------------------------|---------------------------|

### 40:97 MF2-Tastatur ( allgemeiner Tastaturstatus )

|             |                              |                |                                        |                      |                   |                  |                     |
|-------------|------------------------------|----------------|----------------------------------------|----------------------|-------------------|------------------|---------------------|
| Sendefehler | Modus LEDs wird aktualisiert | Empfangsfehler | Bestätigung ACK von Tastatur empfangen | reserviert (stets 0) | Caps-Lock LED ein | Num-Lock LED ein | Scroll-Lock LED ein |
|-------------|------------------------------|----------------|----------------------------------------|----------------------|-------------------|------------------|---------------------|

Bit = 1 bedeutet, daß der entsprechende Modus eingeschaltet bzw. die entsprechende Taste gedrückt ist.

### Die Tastatur-Statusbyte des BIOS

| Wort im Tastaturpuffer       | Oberes Byte | Unteres Byte  |
|------------------------------|-------------|---------------|
| “Normale“ Taste              | Scan-Code   | ASCII-Code    |
| Erweiterte Taste             | Scan-Code   | Null          |
| Eingabe über die “Alt“-Taste | Null        | ASCII-Zeichen |

### Umwandlung eines Make-Code durch das BIOS

| Adresse                                                                                                                           | Größe   | Inhalt                                  | Bedeutung                                                   |
|-----------------------------------------------------------------------------------------------------------------------------------|---------|-----------------------------------------|-------------------------------------------------------------|
| 40:1a                                                                                                                             | Wort    | Lesezeiger                              | zeigt auf das nächste zu lesende Zeichen im Tastaturpuffer  |
| 40:1c                                                                                                                             | Wort    | Schreibzeiger                           | zeigt auf die nächste freie Schreibstelle im Tastaturpuffer |
| 40:1e                                                                                                                             | 32 Byte | Tastaturpuffer                          | 16 Zeichen groß, davon aber nur 15 benutzt                  |
| 40:80                                                                                                                             | Wort    | Beginn des alternativen Tastaturpuffers | Offset ab Segment 0040 h                                    |
| 40:82                                                                                                                             | Wort    | Ende des alternativen Tastaturpuffers   | Offset ab Segment 0040 h                                    |
| Der vom BIOS verwendete, alternative Puffer kann größer als 32 Byte sein, seine Adresse ist aber auf das Segment 0040h beschränkt |         |                                         |                                                             |
| <b>BIOS-Datenbereich des Tastaturpuffers</b>                                                                                      |         |                                         |                                                             |

### BIOS- Tastatur-Hooks mit INT 15h

| Fkt-Nr.                                      | Bedeutung                                                      | Eingabe/Kommentar                                               | Ausgabe/Kommentar                                               |
|----------------------------------------------|----------------------------------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------|
| AH = 4F h<br>für Scancode-<br>Manipulation   | User Exit des Int 09h<br>mit Scancode in AL<br>Aufruf: INT 15h | AH = 4Fh<br>AL = Scancode<br>CY = 1                             | Rückgabe CY = 0 Taste<br>ignorieren<br>CY = 1 Taste verarbeiten |
| AH = 85h<br>für Abfangen von<br>SysReq-Taste | Int 09h ruft Funktion auf<br>wenn Alt+SysReq gedrückt          | AH = 85h<br>AL = 0 SysReq gedrückt<br>AL = 1 SysReq losgelassen | AX = 00<br>CY = 0                                               |

### BIOS – Drucker-Ausgabe mit INT 17h

| Fkt-Nr. | Bedeutung                                              | Eingabe / Kommentar                                                                        | Ausgabe / Kommentar                                               |
|---------|--------------------------------------------------------|--------------------------------------------------------------------------------------------|-------------------------------------------------------------------|
| AH = 0  | Zeichen in AL drucken                                  | DX: log. Drucker<br>00 = LPT1 (PRN)<br>01 = LPT2<br>02 = LPT3<br>AL: zu druckendes Zeichen | AH = 01, falls kein Druck<br>erfolgte,<br>übrigen Bits wie Status |
| AH = 1  | Druckerschnittstelle<br>initialisieren                 | DX: wie oben                                                                               | AH: Status der Schnittstelle<br>nach Rücksprung                   |
| AH = 2  | Status des Druckers bzw<br>der Schnittstelle ermitteln | DX: wie oben                                                                               | AH: Statusinformation                                             |

### Druckerstatus in AH nach Aufruf: AH = 02 INT 17h Bit = 1 : wirksam

|                                |                              |             |                       |                  |                    |                    |                     |
|--------------------------------|------------------------------|-------------|-----------------------|------------------|--------------------|--------------------|---------------------|
| Drucker<br>frei ( Not<br>Busy) | Bestätigung<br>(Acknowledge) | Kein Papier | Drucker<br>ausgewählt | E / A-<br>Fehler | Nicht<br>verwendet | Nicht<br>verwendet | Time-Out-<br>Fehler |
|--------------------------------|------------------------------|-------------|-----------------------|------------------|--------------------|--------------------|---------------------|

### Bildschirmfunktionen von INT 10 h

| Funktions-<br>Nummer | Bedeutung                                                                          | Eingabedaten / Kommentar                                                                                                                                                                                                                                                                                                                                                    | Ausgabedaten /Kommentar                                                                                                                                                                           |
|----------------------|------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AH = 00 h            | Betriebsart wählen<br>Bildschirm wird<br>hierbei gelöscht                          | AL = 00h: Text 40 x 25 Monochrom<br>AL = 01h: Text 40 x 25 Color 16 farbig<br>AL = 02h: Text 80 x 25 Monochrom<br>AL = 03h: Text 80 x 25 Color, 16 farbig, 8<br>AL = 04h: Grafik 320 x 200 Color<br>AL = 05h: Grafik 320 x 200 Monochrom<br>AL = 06h: Grafik 640 x 200 Mono/2 Farben<br>AL = 07h: Monochrom-Bildschirmadapter<br>AL = 12h: Grafik 640x480 16 farbig, 1Seite | keine<br>Bit AL.7 = 1 verhindert beim Aufruf das<br>Löschen des Video-RAMs                                                                                                                        |
| AH = 01 h            | Darstellungsart<br>(Größe) des Bild-<br>schirm-Cursors<br>festlegen                | CH = 0 bis 1F h (b4 ..... b0 )<br>Cursor-Anfangszeile<br>CL = 0 bis 1F h (b4 .....b0 )<br>Cursor-Endzeile                                                                                                                                                                                                                                                                   | keine                                                                                                                                                                                             |
| AH = 02 h            | Cursor auf<br>Bildschirm<br>positionieren                                          | DH: Zeilen-Nr. entsprechend Text oder<br>DL: Spalten-Nr. Grafik-Modus<br>BH: Bildschirmseite, im Grafikmode BH=0                                                                                                                                                                                                                                                            | keine<br>Die Koordinate (0,0) ist in der<br>linken oberen Bildschirmcke                                                                                                                           |
| AH = 03 h            | Position des<br>Cursors bestimmen                                                  | BH: Bildschirmseite, bei Grafik BH=0                                                                                                                                                                                                                                                                                                                                        | DH: Zeilen-Nr.<br>DL: Spalten Nr.<br>CH=0 bis 0Fh Cursor-Anfangszeile<br>CL=0 bis 0Fh Cursor-Endzeile                                                                                             |
| AH = 04 h            | Position eines Licht-<br>stiftes (Light Pen )<br>mit optischen Sensor<br>ermitteln | keine<br>diese Funktion setzt einen Lichtstift voraus,<br>der am Bildschirmadapter angeschlossen<br>ist                                                                                                                                                                                                                                                                     | AH=0: Lichtstift nicht aktiv<br>AH=1:Folgende Positionskoordinaten<br>sind gültig<br>DH: Zeilen-Nr. DL: Spalten-Nr.<br>CH: Grafikzeilen-Nr. (0 bis 199 )<br>BX: Grafikspalten-Nr.(0 bis 319/639 ) |
| AH = 05 h            | Auswählen einer<br>Bildschirmseite                                                 | AL: Bildschirmseite<br>(0 bis 7, je nach Modus )                                                                                                                                                                                                                                                                                                                            | keine Bei Grafik wird der gesamte Bild-<br>wiederholpeicher benutzt, daher nur<br>gültig für Textmodi                                                                                             |
| AH = 06 h            | Fenster auf aktueller<br>Seite nach oben<br>rollen (Scrolling Up)                  | AL: Zahl der Scroll-Zeilen<br>CH: Zeilen-/<br>CL: Spaltenposition des Scroll-Fensters<br>links oben<br>DH: Zeilen-/Spaltenposition des Scroll-<br>DL: Fensters rechts unten<br>BH: Bildschirmattribut für nachgeschobene<br>Leerzeilen                                                                                                                                      | keine<br>Die Bildschirmattribute sind auf<br>Seite definiert<br><br>Bildschirm löschen:<br>AL=0 BH=Attribut<br>CH=0 CL=0 DH=24 DL=79                                                              |
| AH = 07 h            | Fenster auf aktueller<br>Seite nach unten rol-<br>len (Scrolling Down)             | AL: Zahl der Scroll-Zeilen<br>CH: Zeilen-/<br>CL: Spaltenposition des Scroll-Fensters<br>links oben<br>DH: Zeilen-/Spaltenposition des Scroll-<br>Fensters rechts unten<br>BH: Bildschirmattribut für nachgeschobene<br>Leerzeilen                                                                                                                                          | keine<br>Die Attributbits sind kompatibel<br>bei Monochrom-/Color-Adaptern                                                                                                                        |
| AH = 08 h            | Bestimmen eines<br>Zeichens mit Attribut<br>an der Cursorposition                  | BH: Bildschirmseite<br>( nur bei Textmodi )                                                                                                                                                                                                                                                                                                                                 | AL: gelesenes Zeichen ASCII<br>AH: gelesenes Bildschirmattribut<br>( nur Textmodi )                                                                                                               |
| AH = 09 h            | Schreiben von<br>Zeichen mit<br>Attribut an die<br>Cursorposition                  | AL: zu schreibendes Zeichen ASCII<br>BH: Bildschirmseite (nur Textmodi)<br>BL: Attribut (Textmodi) oder<br>Farbe ( Grafikmodi )<br>CX: Wiederholungsfaktor Anzahl Zeichen                                                                                                                                                                                                   | keine<br>der Wiederholungsfaktor gilt nur für laufende<br>Zeile (kein Zeilenumbruch)<br>Cursor muß separat bedient werden (Fktn 2 )<br>Attribut werden verändert                                  |

|          |                                                     |                                                                                                                                                                                             |                                                                                                                                                                     |
|----------|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AH = 0Ah | Schreiben eines Zeichens an aktuelle Cursorposition | AL: zu schreibendes Zeichen ASCII<br>BH: Bildschirmseite (nur Textmodi)<br>BL: Zeichenfarbe (Grafik)<br>CX: Wiederholungsfaktor                                                             | keine<br>der Wiederholungsfaktor gilt nur für laufende Zeile (kein Zeilenumbruch)<br>Cursor muß separat bedient werden (Fktn 2)<br>Attribute werden nicht verändert |
| AH = 0Bh | Farbpalette festlegen                               | BH: Nummer der Farbpalette (0 .... 7F h)<br>BL: Farbwert für diese Farbpalette (bei CGA nur Modus 4)                                                                                        | Keine<br>ist BH=0 so enthält BL die Hintergrund-/Randfarbe (00 .... 0F h)<br>BH = 01h; BL enthält die Paletten Nr. 0..1                                             |
| AH = 0Ch | Grafikpunkt schreiben                               | DX: Zeilen-Nr. Y-Koordinate<br>CX: Spalten-Nr. X-Koordinat<br>AL: Farbe (Bitstelle 0 bis 6)<br>Bit b7 = 1; Exklusiv-ODER-Verknüpfung mit der aktuellen Pixelfarbe<br>BH = 0 Bildschirmseite | Keine<br>über die XOR-Verknüpfung kann eine Bildung der komplementären Farbe erreicht werden                                                                        |
| AH = 0Dh | lesen eines Grafikpunktes                           | DX: Zeilen-Nr. BH = 0 Bildschirmseite<br>CX: Spalten-Nr.                                                                                                                                    | AL: Farbe des gewählten Punktes                                                                                                                                     |
| AH = 0Eh | Zeichenausgabe mit Weiterpositionierung des Cursors | AL: auszugebendes Zeichen<br>BL: Zeichenfarbe                                                                                                                                               | Keine                                                                                                                                                               |
| AH = 0Fh | Abfrage des aktuellen Bildschirmstatus              | keine                                                                                                                                                                                       | AL: Bildschirmmodus (wie Funktion AH = 0)<br>AH: Zahl der Bildschirmspalten (40 o. 80)<br>BH: aktuelle Bildschirmseite                                              |
| AH = 13h | Ausgabe einer Zeichenkette                          | AL: Ausgabemodus<br>CX: Anzahl Zeichen<br>BH: Bildschirmseite<br>BL: Attribut-Byte<br>DX: Zeile-Spalte<br>ES: BP Pufferzeiger                                                               | Modi: 0=Attribut BL; Cursor beibehalten<br>1=Attribut BL; Cursor aktualisiert<br>2=Attribut Puffer; Cursor beibehalten<br>3=Attribut Puffer; Cursor aktualisiert    |

**Aufbau des Text-Attributes (Mischfarben) auch für den Text-Bildspeicher:**

|                                       |     |      |      |                         |     |      |      |
|---------------------------------------|-----|------|------|-------------------------|-----|------|------|
| <----- Bildfarbe (Hintergrund) -----> |     |      |      | <----- Textfarbe -----> |     |      |      |
| Bit 7                                 | rot | grün | blau | hell                    | rot | grün | blau |

Bit 7: Textblinken bzw. Bildhell je nach Treiber

| Farbwerte für Pixelgrafik mit Helligkeitsbit (für Grafik-Modus 12h 640x480 16 farbig) |           |                |               |            |              |                  |                   |
|---------------------------------------------------------------------------------------|-----------|----------------|---------------|------------|--------------|------------------|-------------------|
| 0 = schwarz                                                                           | 4 = rot   | 8 = dunkelgrau | 0Ch = hellrot | 1 = blau   | 5 = fuchsin  | 9 = hellblau     | 0Dh = hellfuchsin |
| 2 = grün                                                                              | 6 = braun | 0Ah = hellgrün | 0Eh = gelb    | 3 = türkis | 7 = hellgrau | 0Bh = helltürkis | 0Fh = weiß        |

**Aufbau des Text-Bildspeichers im Segment: 0B800h**

| 1. Spalte |       |          | 2.Spalte |          | 80. Spalte |       |          |
|-----------|-------|----------|----------|----------|------------|-------|----------|
|           | Code  | Attribut | Code     | Attribut |            | Code  | Attribut |
| 1.Zeile   | +0000 | +0001    | +0002    | +0003    |            | +0158 | +0159    |
| 2.Zeile   | +0160 | +0161    | +0162    | +0163    |            | +0318 | +0319    |
| 3.Zeile   | +0320 | +0321    | +0322    | +0323    |            | +0478 | +0479    |
|           |       |          |          |          |            |       |          |
| 24.Zeile  | +3680 | +3681    | +3682    | +3683    |            | +3838 | +3839    |
| 25.Zeile  | +3840 | +3841    | +3842    | +3843    |            | +3998 | +3999    |

## Interrupt-Routinen für die Maus Aufruf: `int 33h`

| Funktion                                                                          | Aufruf                                                                                                        | Rückgabe                                                                                                                                                                        |
|-----------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| AX = 0 Status lesen                                                               | -                                                                                                             | AX = Maus vorhanden ? = FFFF h => ja<br>BX = Zahl der Tasten 0000 h => nein                                                                                                     |
| AX = 1 Maus-Zeiger ein                                                            | -                                                                                                             | Standard (Textmode) -> Softwarezeiger Block                                                                                                                                     |
| AX = 2 Maus-Zeiger aus                                                            | -                                                                                                             | -                                                                                                                                                                               |
| AX = 3 Mauszustand aktueller Status der Tasten und Position des Mauszeigers       | -                                                                                                             | BX = Tasten (xxxxx mitte=1 rechte=1 linke=1)<br>CX = X-Position in Pixel<br>DX = Y-Position in Pixel                                                                            |
| AX = 4 Mauszeigerposition festlegen                                               | CX = X-Position in Pixel<br>DX = Y-Position in Pixel                                                          |                                                                                                                                                                                 |
| AX = 5 Taste gedrückt? Betätigungen einer Maustaste und letzte Mauszeigerposition | BX = Tastencode<br>0:linke 1:rechte 2:mitte<br>(Code je nach Treiber)                                         | AX =Tasten (.....mitte rechte linke)<br>BX = Anzahl der Tasten-Betätigungen seit letzter Abfrage<br>CX = X-Position in Pixel bei letzter<br>DX = Y-Position in Pixel Betätigung |
| AX = 6 Taste gelöst? Freigeben einer Maustaste und letzte Mauszeigerposition      | BX = Tastencode<br>0:linke 1:rechte 2:mitte<br>(Code je nach Treiber)                                         | AX =Tasten (.....mitte rechte linke)<br>BX = Anzahl der Tasten -Freigaben<br>CX = X-Position in Pixel bei letzter<br>DX = Y-Position in Pixel Freigabe                          |
| AX = 7 Horizontalbereich einstellen                                               | CX = linke Grenze<br>DX = rechte Grenze                                                                       | Fenster für Mauszeiger festlegen<br>z.B. (0..639 Pixel)                                                                                                                         |
| AX = 8 Vertikalbereich einstellen                                                 | CX = obere Grenze<br>DX = untere Grenze                                                                       | Fenster für Mauszeiger festlegen<br>z.B. (0..479 Pixel)                                                                                                                         |
| AX = 9 Mauszeiger im Grafikmodus definieren                                       | BX = Aktionspunkt horizontal<br>CX = Aktionspunkt vertikal<br>ES: DX= FAR-Zeiger                              | Wertebereich -16 ... +16<br>Adresse der Maskenpuffer Screen-Cursor                                                                                                              |
| AX =0ah Mauszeiger im Textmodus definieren                                        | BX = Mauszeigertyp →<br>CX = Screen-Maske UND<br>DX = Cursor-Maske XOR                                        | 0 = Software-Mauszeiger (UND / XOR-Maske)<br>1 = Hardware-Mauszeiger = Cursor<br>CX/DX= Start/Endzeile Zeiger/Cursor-Blocks                                                     |
| AX =0bh Bewegungszähler der Maus lesen                                            | CX = Zählwert horizontal<br>DX= Zählwert vertikal →<br>seit letztem Aufruf                                    | 1 Zählwert ( Mickey ) = 0,13 mm                                                                                                                                                 |
| AX =0fh Verhältnis Zählwert / Bildschirmpunkt festlegen                           | CX = horizontales Verhältnis<br>DX= vertikales Verhältnis                                                     | Wertebereich 1... 32767<br>Standard: horizontal = 8 , vertikal = 16                                                                                                             |
| AX =10h Mauszeiger bedingt ausschalten                                            | CX = X-Wert rechte Grenze<br>DX= Y-Wert untere Grenze<br>SI = X-Wert linke Grenze<br>DI = Y-Wert obere Grenze | definiert ein Fenster, innerhalb dessen der Mauszeiger gelöscht wird<br>für erneute Darstellung Funktion 01                                                                     |
| AX =13h Schwellenwert für doppelte Geschwindigkeit definieren                     | DX= Schwellenwert                                                                                             | legt Schwellenwert in Zählwerte (Mickey) /sec fest, ab dem Mauszeiger mit doppelter Geschwindigkeit auf Bildschirm bewegt wird<br>Standardschwellenwert = 64                    |

### BIOS-Funktionen für Datum und Uhrzeit im PC ( DOS-Systemuhr und Echtzeituhr RTC )

| Aufruf          | Funktion                                                                                                                                                                                                 | Eingabe                                                                                                                            | Rückgabe                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| INT 1ah<br>BIOS | Zeitähler lesen DOS-interne Systemuhr<br>Zahl der Timer-Ticks seit Einschalten des<br>PCs 0:00 Uhr gezählt in <b>40:6fh - 40:6ch</b><br><b>40:70h</b> = 1 > 24-Std-Grenze                                | AH = 0                                                                                                                             | AL = 24-Stunden-Angabe<br>CX = Zählerstand high<br>DX = Zählerstand low                     |
| INT 1ah<br>BIOS | Zeitähler setzen DOS-interne Systemuhr<br>Setzen der Timer-Ticks seit 0:00 Uhr                                                                                                                           | AH = 01<br>CX = Zähler- high<br>DX= Zähler-low                                                                                     |                                                                                             |
| INT 1ah<br>BIOS | Uhrzeit lesen – Echtzeituhr                                                                                                                                                                              | AH = 02                                                                                                                            | AH = 00<br>CL = Minute BCD<br>CH = Stunde BCD<br>DH = Sekunde BCD<br>CY = 1 -> Fehler       |
| INT 1ah<br>BIOS | Uhrzeit setzen – Echtzeituhr                                                                                                                                                                             | AH = 03<br>CL = Minute BCD<br>CH = Stunde BCD<br>DL = 1 Sommerzeit 0=nein<br>DH = Sekunde BCD                                      | AH = 00<br><br>CY = 1 -> Fehler                                                             |
| INT 1ah<br>BIOS | Datum lesen – Echtzeituhr                                                                                                                                                                                | AH = 04                                                                                                                            | CL = Jahr BCD<br>CH = Jahrhundert BCD<br>DL = Tag BCD<br>DH = Monat BCD<br>CY = 1 -> Fehler |
| INT 1ah<br>BIOS | Datum setzen – Echtzeituhr                                                                                                                                                                               | AH = 05<br>CL = Jahr BCD<br>CH = Jahrhundert BCD<br>DL = Tag BCD<br>DH = Monat BCD                                                 | CY = 1 -> Fehler                                                                            |
| INT 1ah<br>BIOS | Alarmzeit setzen – Echtzeituhr<br>bei Erreichen der Alarmzeit wird ein<br>Interrupt 4ah ausgelöst                                                                                                        | AH = 06<br>CL = Minute BCD<br>CH = Stunde BCD<br>DH = Sekunde BCD                                                                  | CY = 1 -> Fehler                                                                            |
| INT 1ah<br>BIOS | Alarmzeit löschen – Echtzeituhr<br>(vor Setzen einer neuen Alarmzeit )                                                                                                                                   | AH = 07                                                                                                                            | CY = 1 -> Fehler                                                                            |
| INT 15h<br>BIOS | Wartezeit setzen<br>nach Ablauf der Wartezeit löst Echtzeituhr<br>Interrupt aus Zeiten in µsec<br>zeitliche Auflösung aber bei 1/1024 Hz<br>Bit 7 des Zielbyte wird nach Ablauf der<br>Wartezeit gesetzt | AH = 83h<br>AL = 00<br>CX = Zeitintervall high<br>DX = Zeitintervall low<br>BX = Offset des Zielbytes<br>ES = Segment des Zielbyte | AH = 00<br>AL = Statusregister B<br><br>CY = 1 -> Fehler                                    |
| INT 15h<br>BIOS | Wartezeit löschen                                                                                                                                                                                        | AH = 83h<br>AL = 01h                                                                                                               |                                                                                             |
| INT 15h<br>BIOS | Zeitintervall abwarten<br>nach Ablauf des Intervalls wird die Aus-<br>führung des aufrufenden Programmes<br>fortgesetzt                                                                                  | AH = 86h<br>CX = Zeitintervall high<br>DX = Zeitintervall low<br>Zeit in µsec                                                      | CY = 1 -> Fehler                                                                            |

## DOS-Application-Program-Interface

Zu den sog. DOS-Interrupts zählen die Interrupt-Nr.: 20 h – 2F h. Der wichtigste ist der **INT 21 h – Aufruf einer DOS-Funktion**. Über den Interrupt 21h können mehr als 100 Funktionen erreicht werden, die das DOS einem Programm zur Verfügung stellt und die deshalb als Application-Program-Interface (DOS-API) bezeichnet werden.

### Übersicht der Funktionen des Interrupts 21h

#### Zeicheneingabe

|     |                                                        |
|-----|--------------------------------------------------------|
| 01h | Zeicheneingabe mit Ausgabe                             |
| 03h | Empfang eines Zeichens von der seriellen Schnittstelle |
| 06h | Direkte Zeichenein-/ausgabe                            |
| 07h | Direkte Zeicheneingabe ohne Ausgabe                    |
| 08h | Zeicheneingabe ohne Ausgabe                            |
| 0Ah | Eingabe einer Zeichenkette                             |
| 0Bh | Lese Eingabestatus                                     |
| 0Ch | Lösche Eingabepuffer und rufe Eingabefunktion auf      |

#### Zeichenausgabe

|     |                                                       |
|-----|-------------------------------------------------------|
| 02h | Ausgabe eines Zeichens                                |
| 04h | Ausgabe eines Zeichens auf die serielle Schnittstelle |
| 05h | Ausgabe auf parallele Schnittstelle                   |
| 06h | Direkte Zeichenein-/ausgabe                           |
| 09h | Ausgabe einer Zeichenkette                            |

#### Programmbeendigung

|     |                                             |
|-----|---------------------------------------------|
| 00h | Programm beenden                            |
| 31h | Programm beenden, aber im Speicher belassen |
| 4Ch | Programm mit Ende-Code beenden              |

#### Zugriff auf Unterverzeichnisse

|     |                                 |
|-----|---------------------------------|
| 39h | Unterverzeichnis erstellen      |
| 3Ah | Unterverzeichnis löschen        |
| 3Bh | Aktuelles Verzeichnis setzen    |
| 47h | Aktuelles Verzeichnis ermitteln |

#### RAM-Speicher-Verwaltung

|         |                                       |
|---------|---------------------------------------|
| 48h     | RAM-Speicher reservieren              |
| 49h     | RAM-Speicher freigeben                |
| 4Ah     | Größe eines Speicherbereichs ändern   |
| 58h/00h | Konzept der Speicherverteilung lesen  |
| 58h/01h | Konzept der Speicherverteilung setzen |
| 58h/02h | Einbindung der UMBs abfragen          |
| 58h/03h | Einbindung der UMBs festlegen         |

#### Zugriff auf Gerätetreiber

|           |                                                   |
|-----------|---------------------------------------------------|
| 44h/00h   | IOCTL: Lesen des Geräte-Attributs                 |
| 44h/01h   | IOCTL: Setzen des Geräte-Attributs                |
| 44h/02h   | IOCTL: Daten von einem Zeichentreiber empfangen   |
| 44h/03h   | IOCTL: Daten an einen Zeichentreiber senden       |
| 44h/04h/1 | IOCTL: Daten von einem Blocktreiber empfangen     |
| 44h/04h/2 | DBLSPC: Internen Cache schreiben                  |
| 44h/04h/3 | DBLSPC: Internen Cache schreiben und invalidieren |
| 44h/05h   | IOCTL: Daten an einen Blocktreiber übertragen     |
| 44h/06h   | IOCTL: Eingabestatus abfragen                     |
| 44h/07h   | IOCTL: Ausgabestatus abfragen                     |
| 44h/08h   | IOCTL: Ist das Medium wechselbar?                 |
| 44h/09h   | IOCTL: Device-Remote-Test                         |
| 44h/0Ah   | IOCTL: Handle-Remote-Test                         |
| 44h/0Bh   | IOCTL: Zugriffswiederholung setzen                |
| 44h/0Ch   | IOCTL: Kommunikation mit einem Zeichentreiber     |
| 44h/0Dh   | IOCTL: Kommunikation mit einem Blocktreiber       |
| 44h/0Eh   | IOCTL: Letzte Laufwerksbezeichnung ermitteln      |
| 44h/0Fh   | IOCTL: Nächste Laufwerksbezeichnung definieren    |

- 44h/10h IOCTL-Unterstützung auf Handle-Ebene abfragen
- 44h/11h IOCTL-Unterstützung auf Geräte-Ebene abfragen

### **Uhrzeit und Datum**

- 2Ah Datum abfragen
- 2Bh Datum setzen
- 2Ch Uhrzeit abfragen
- 2Dh Uhrzeit setzen

### **Diskettenübertragungsbereich**

- 1Ah Setzen der DTA-Adresse
- 2Fh DTA ermitteln

### **Directory durchsuchen**

- 11h Suche ersten Directory-Eintrag (FCB)
- 12h Suche nächsten Directory-Eintrag (FCB)
- 4Eh Ersten Directory-Eintrag suchen (Handle)
- 4Fh Nächsten Directory-Eintrag suchen (Handle)

### **Dateizugriff (FCB)**

- 0Fh Datei öffnen (FCB)
- 10h Datei schließen (FCB)
- 13h Datei(en) löschen (FCB)
- 14h Sequentielles Lesen (FCB)
- 15h Sequentielles Schreiben (FCB)
- 16h Erstellen oder Leeren einer Datei (FCB)
- 17h Datei(en) umbenennen (FCB)
- 21h Wahlfreies Lesen (FCB)
- 22h Wahlfreies Schreiben (FCB)
- 23h Lese Dateigröße (FCB)
- 24h Setze Datensatznummer
- 27h Wahlfreies Lesen mehrerer Datensätze (FCB)
- 28h Wahlfreies Schreiben mehrerer Datensätze (FCB)
- 29h Dateinamen in FCB übertragen

### **Dateizugriff (Handle)**

- 3Ch Datei erstellen oder leeren (Handle)
- 3Dh Datei öffnen (Handle)
- 3Eh Datei schließen (Handle)
- 3Fh Datei lesen (Handle)
- 40h Datei beschreiben (Handle)
- 41h Datei löschen (Handle)
- 42h Dateizeiger bewegen (Handle)
- 45h Handle verdoppeln
- 46h Handles angleichen
- 56h Datei umbenennen oder verschieben (Handle)
- 5Ah Temporäre Datei erstellen (Handle)
- 5Bh Neue Datei erstellen (Handle)
- 5Ch/00h Bereich einer Datei gegen Zugriff schützen
- 5Ch/01h Freigabe eines gesperrten Bereichs in einer Datei
- 6Ch Erweiterte OPEN-Funktion

### **Netzwerk-Aufrufe**

- 5Eh/00h Namen des Rechners im Netzwerk ermitteln
- 5Eh/02h Initialisierungs-String für Netzwerkdrucker festlegen
- 5Eh/03h Initialisierungs-String für Netzwerkdrucker ermitteln
- 5Fh/02h Eintrag aus der Netzwerkliste holen
- 5Fh/03h Eintrag in der Netzwerkliste definieren
- 5Fh/04h Eintrag aus der Netzwerkliste entfernen

### **Zugriff auf Interrupt-Vektoren**

- 25h Setze Interrupt-Vektor
- 35h Inhalt eines Interrupt-Vektors auslesen

### **Zugriff auf Disketten/Festplatten**

|     |                                                      |
|-----|------------------------------------------------------|
| 0Dh | Reset der Blocktreiber                               |
| 0Eh | Auswahl des aktuellen Laufwerks                      |
| 19h | Gerätebezeichnung des aktuellen Laufwerks erfragen   |
| 1Bh | Informationen über das aktuelle Laufwerk einholen    |
| 1Ch | Informationen über ein beliebiges Laufwerk einholen  |
| 1Fh | DPB-Zeiger für das aktuelle Laufwerk ermitteln       |
| 32h | Zeiger auf DPB für ein beliebiges Laufwerk ermitteln |
| 36h | Verbleibende Plattenkapazität ermitteln              |
| 53h | BPB in DPB umsetzen                                  |

### **Zugriff auf den PSP**

|     |                           |
|-----|---------------------------|
| 26h | Erstelle neuen PSP        |
| 50h | Aktiven PSP setzen        |
| 51h | Aktiven PSP ermitteln     |
| 55h | Neuen PSP erstellen       |
| 62h | Adresse des PSP ermitteln |

### **Zugriff auf DOS-Flags**

|         |                                                   |
|---------|---------------------------------------------------|
| 2Eh     | Setzen des Verify-Flags                           |
| 33h/00h | Lesen des Break-Flags                             |
| 33h/01h | Setzen des Break-Flags                            |
| 34h     | DOS Zeiger auf das INDOS-Flag ermitteln           |
| 37h/00h | Kennzeichen für Kommandozeilen-Schalter ermitteln |
| 37h/01h | Kennzeichen für Kommandozeilen-Schalter setzen    |
| 52h     | Zeiger auf DOS-Info-Block ermitteln               |
| 54h     | Verify-Flag lesen                                 |

### **Zugriff auf Datei-Informationen**

|         |                                                                  |
|---------|------------------------------------------------------------------|
| 43h/00h | Attribut einer Datei ermitteln                                   |
| 43h/01h | Attribut einer Datei setzen                                      |
| 57h/00h | Datum und Uhrzeit der letzten Modifikation einer Datei ermitteln |
| 57h/01h | Datum und Uhrzeit der letzten Modifikation einer Datei setzen    |

### **Zugriff auf landesspezifische Parameter**

|         |                                                 |
|---------|-------------------------------------------------|
| 38h     | Landesspezifische Symbole und Formate ermitteln |
| 38h/00h | Landesspezifische Symbole und Formate ermitteln |
| 38h/01h | Land setzen                                     |

### **Verschiedene Funktionen**

|         |                                          |
|---------|------------------------------------------|
| 30h     | DOS-Versionsnummer ermitteln             |
| 4Bh/00h | EXEC: anderes Programm ausführen         |
| 4Bh/03h | EXEC: anderes Programm als Overlay laden |
| 4Bh/05h | EXEC: eigene EXECs anpassen              |
| 4Dh     | Ende-Code ermitteln                      |
| 59h     | Erweiterte Fehlerinformationen einholen  |
| 60h     | Dateinamen erweitern                     |
| 66h/01h | Aktuelle Code-Page ermitteln             |
| 66h/02h | Aktuelle Code-Page festlegen             |
| 67h     | Anzahl der verfügbaren Handles festlegen |
| 68h     | Dateipuffer leeren                       |

**DOS-Interrupt-Routinen** für die **Ein- / Ausgabe** von Zeichen und Strings auf der **Konsole**:

| Aufruf  | Funktion                                  | Aufgabe                                                                                                                                                                                                                                                                                                                             |
|---------|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| int 21h | AH = 1                                    | Tastatur-Eingabe nach <b>AL</b> mit Warten und Echo ( Ctrl C -> INT 23h )<br>gefiltert -> Steuerzeichen ( cr, lf, bcksp, usw. ) funktionieren<br>Erweiterte Tastaturcodes: zuerst 0, erneuter Aufruf -> eigentlicher Code                                                                                                           |
| int 21h | AH = 2                                    | Zeichen aus <b>DL</b> auf dem Bildschirm ausgeben ; AL wird verändert<br>gefiltert, dh. Steuerzeichen funktionieren                                                                                                                                                                                                                 |
| int 21h | AH = 06<br>DL= 0 -254<br>DL= 255          | Zeichen- <b>Eingabe</b> ( ohne Warten ohne Echo ) und Zeichen- <b>Ausgabe</b><br>dieses ASCII – Zeichen ausgeben ( ASCII-Code 255 nicht möglich )<br>Ein Zeichen von Tastatur <b>ohne Warten und ohne Echo</b> nach AL ein<br>Z-Flag = 1 : kein Zeichen bereit; Z-Flag = 0 : Zeichen in AL                                          |
| int 21h | AH = 7                                    | Tastatur-Eingabe nach <b>AL</b> mit Warten ohne Echo<br>(ungefiltert d.h. z.B. Ctrl C keine Wirkung)                                                                                                                                                                                                                                |
| int 21h | AH = 8                                    | Tastatur-Eingabe nach <b>AL</b> mit Warten ohne Echo ( Ctrl C -> INT 23h )<br>gefiltert, erweiterte Tastaturcodes: zuerst 0, dann eigentlicher Code                                                                                                                                                                                 |
| int 21h | AH = 09<br>DS:DX =                        | Ausgabe einer Zeichenkette Zeichenkettenende muß '\$'-Zeichen sein<br>FAR-Zeiger auf Beginn der Zeichenkette ; AL wird verändert<br>gefiltert, Steuerzeichen funktionieren und Ctrl C -> INT 23h                                                                                                                                    |
| int 21h | AH = 0Ah<br>DS:DX =                       | Eingabe einer <b>Zeichenkette mit Echo</b> ( Ctrl C -> INT 23h )<br>FAR-Zeiger auf Puffer fuer Zeichenkette<br>1.Pufferbyte muß max. Anzahl der einzugebenden Zeichen enthalten<br>2.Byte enthaelt Anzahl der tatsächlich eingegeben Zeichen ohne cr<br>Funktion mit cr beenden - cr wird als letztes Zeichen im Puffer eingetragen |
| int 21h | AH = 0Bh                                  | Tastatur-Test <b>ohne Warten</b> (sofort zurück!!!)<br>AL = 0: keine Taste gedrückt<br>AL = 255: Taste gedrückt, Zeichen im Puffer !<br>abholen mit AH = 1 oder AH = 8                                                                                                                                                              |
| int 21h | AH = 0Ch<br>AL =                          | Lösche Tastatur-Eingabepuffer und rufe eine Eingabefunktion auf<br>Nummer der aufzurufenden Funktion (1, 6, 7, 8, 0Ah) ; AL verändert                                                                                                                                                                                               |
| int 21h | AH = 3Fh<br>BX =handle<br>CX =<br>DS:DX = | Datei lesen nach Pufferspeicher ( bei Tastatur mit Echo )<br>0 = handle für Tastatur CON Rückgabe: AX = Anzahl gelesener Byte<br>Anzahl der Bytes CY=0 o.k.<br>Adresse des Pufferspeichers CY = 1 Fehler, in AX = Fehlercode<br>(Standard unter DOS ist cooked-Mode = Steuerzeichen z.B. <cr> aktiv)                                |
| int 21h | AH = 40h<br>BX =handle<br>CX =<br>DS:DX = | Datei beschreiben aus Pufferspeicher<br>1 = handle für Bildschirm Rückgabe: AX = Anzahl geschriebener Byte<br>Anzahl der Bytes CY=0 o.k.<br>Adresse des Pufferspeichers CY = 1 Fehler, in AX = Fehlercode                                                                                                                           |

### DOS-Funktionen für Interrupt-Vektor Aufruf: INT 21h

| Fkt-Nr | Bedeutung                      | Eingabedaten / Kommentar                                            | Ausgabedaten / Kommentar                          |
|--------|--------------------------------|---------------------------------------------------------------------|---------------------------------------------------|
| 25 h   | Setzen eines Interrupt-Vektors | AH = 25 h<br>AL = Interrupt-Nummer<br>DS:DX Zeiger Seg :Off auf ISR | keine Ausgabe                                     |
| 35 h   | Holen eines Interrupt-Vektors  | AH = 35 h<br>AL = Interrupt-Nummer                                  | ES : BX Vektoradresse<br>Segment : Offset der ISR |

### DOS-Funktionen für Programm-Beendigung Aufruf: INT 21h

| Fkt-Nr | Bedeutung                                  | Eingabedaten / Kommentar                                                                       | Ausgabedaten / Kommentar                                       |
|--------|--------------------------------------------|------------------------------------------------------------------------------------------------|----------------------------------------------------------------|
| 31 h   | Programm beenden aber im Speicher belassen | AH = 31 h AL = Ende-Code<br>DX = Anzahl reservierende Paragraphen (16 Byte) einschließlich PSP | keine Ausgabe für TSR-Programme (Terminate and stay resident ) |
| 4C h   | Programm mit Ende-Code beenden             | AH = 4C h<br>AL = Ende-Code                                                                    | Keine Ausgabe<br>Zuverlässigste Beendigung                     |

### Undokumentierte Funktionsaufrufe Aufruf: INT 21h

| Funktion                                     | Eingabe                                      | Rückgabe                                                                                                |
|----------------------------------------------|----------------------------------------------|---------------------------------------------------------------------------------------------------------|
| Holen Programmsegment Präfix-Adresse ( PSP)  | AH = 62 h                                    | BX= Segmentadresse des aktuellen PSP                                                                    |
| Setzen Programmsegment-Präfixadresse (PSP)   | AH = 50 h BX = Segment-<br>adresse neuer PSP | keine                                                                                                   |
| DOS-Infoblock ermitteln                      | AH = 52 h                                    | ES:BX = Zeiger auf DOS-Infoblock                                                                        |
| Holen des Reentry-Flags<br><b>INDOS-Flag</b> | AH = 43 h                                    | ES:BX Byte-Adresse des INDOS-Flags<br>= 0: keine DOS-Funktion aktiv<br>!= 0: DOS-Funktion in Ausführung |

### Aufbau des DOS-Info-Blocks ( DIB )

| Offset<br>adresse | Länge /<br>Bytes | Bedeutung                                |
|-------------------|------------------|------------------------------------------|
| -04h              | 4                | Zeiger auf 1. Speichersteuerblock ( MCB) |
| 00h               | 4                | Zeiger auf DOS-Parameterblock            |
| 04h               | 4                | Zeiger auf letztbenutzten Puffer         |
| 08h               | 4                | Zeiger auf CLOCK-Einheitentreiber        |
| 0Ch               | 4                | Zeiger auf CON-Einheitentreiber          |
| 10h               | 2                | Maximale Sektorlänge                     |
| 12h               | 4                | Zeiger auf Puffer                        |
| 16h               | 4                | Zeiger auf Redirector-Table              |
| 1Ah               | 4                | Zeiger auf Dateitabelle                  |

### DOS-Funktionen für Datum und Uhrzeit im PC ( DOS-interne Systemuhr ) INT 21h

| Funktion                                                                      | Eingabe                                                            | Rückgabe                                                 |
|-------------------------------------------------------------------------------|--------------------------------------------------------------------|----------------------------------------------------------|
| Datum abfragen Werte dual zurückgeliefert (Wochentag 0=Sonntag 1=Montag ... ) | AH = 2Ah                                                           | AL = Tag der Woche<br>CX = Jahr DH = Monat DL = Tag      |
| Datum setzen<br>Werte dual setzen                                             | AH = 2Bh CX = Jahr<br>DH = Monat DL = Tag                          | AL = 0 -> o.k.<br>AL = 255 -> unplausibel                |
| Uhrzeit abfragen<br>Werte dual                                                | AH = 2Ch                                                           | CH = Stunde CL=Minute<br>DH = Sekunde DL = 1/100 Sekunde |
| Uhrzeit setzen<br>Werte dual                                                  | AH = 2Dh CH = Stunde<br>CL = Minute DH = Sek<br>DL = 1/100 Sekunde | AL = 0 -> o.k.<br>AL = 255 -> unplausibel                |

## DOS-Funktionen für Verwaltung des Arbeitsspeichers ( RAM) Aufruf : INT 21h

Mit dem DOS-Befehl **mem /d /more** läßt sich die Speicherbelegung darstellen

| Funktion                                                                                       | Eingabe                                                                                                                  | Rückgabe                                                                                                                   |
|------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------|
| Reservieren von Arbeitsspeicher<br>Allocate Memory                                             | AH = 48h<br>BX = benötigte Speichergröße in Paragraphen ( 16Byte )                                                       | CY = 0 erfolgreich<br>AX = Anfangssegment<br>CY = 1 AX = Fehlernummer<br>BX = Größe des größten verfügbaren Speicherblocks |
| Freigeben von reserviertem Arbeitsspeicher ( Free Allocate Memory ) – reserviert zuvor mit 48h | AH = 49h<br>ES = Segment des Blockes, der freigegeben werden soll                                                        | CY = 0 o.k.<br>CY = 1 AX = Fehlernummer<br>07->MCBs zerstört                                                               |
| Verändern des reservierten Speicher blocks (SETBLOCK)                                          | AH = 4Ah<br>BX = neue angeforderte Blockgröße in Paragraphen<br>ES = Segment des Speicherblocks                          | CY = 0 o. k.<br>CY = 1 AX = Fehlernummer<br>BX = Größe des größten verfügbaren Speicherblock                               |
| Prüfen / Verändern der Speicher-Reservierungsstrategie                                         | AH = 58 h AL = 0 Prüfen AL = 1 Verändern<br>BX = Strategiemodus 00=Tiefenanpas<br>01=beste Anpassung<br>02=Hochanpassung | CY = 0 o.k. AX=Strategiemodus<br>CY = 1 AX = Fehlernummer<br>01= ungültiger Funktionscode                                  |

## Speicherverwaltung unter DOS

erfolgt mit Hilfe von Speicher-Kontrollblocks ( **Memory Control Blocks - MCB** ) der Größe von 16 Byte, die dem allokierten Speicherbereich im transienten Programm-Bereich ( TPA ) vorangestellt werden.

### Struktur eines Speicher-Kontrollblocks (MCB)

| 0                                                                                             | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8            | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-----------------------------------------------------------------------------------------------|---|---|---|---|---|---|---|--------------|---|----|----|----|----|----|----|
|                                                                                               |   |   |   |   |   |   |   | Programmname |   |    |    |    |    |    |    |
| nicht benutzt                                                                                 |   |   |   |   |   |   |   |              |   |    |    |    |    |    |    |
| Länge in Paragraphen (16 Byte)                                                                |   |   |   |   |   |   |   |              |   |    |    |    |    |    |    |
| Besitzer = Segmentadresse des zugehörigen PSP ( nur von Bedeutung bei Umgebung= Environment ) |   |   |   |   |   |   |   |              |   |    |    |    |    |    |    |
| Verkettungszustand : 'M' = 4Dh -> es folgen weitere MCB ----- 'Z' = 5Ah -> letzter MCB        |   |   |   |   |   |   |   |              |   |    |    |    |    |    |    |

Unmittelbar hinter dem Ende des allokierten Speicherbereiches folgt der nächste MCB, damit zeigt die ' Länge ' auch die Entfernung zum nächsten MCB -1 (verkettete Liste ) an.

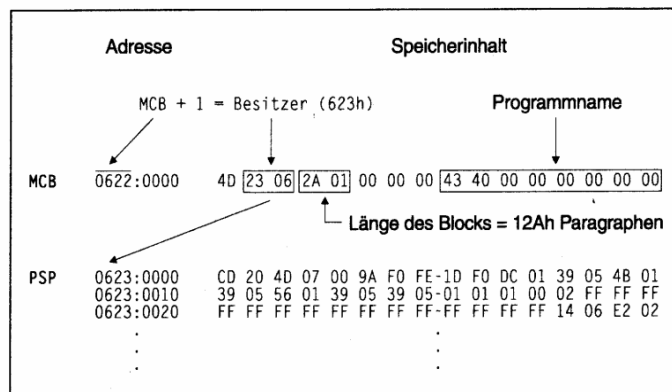
Adresse des ersten MCB im DIB ( DOS Information Block ). Mit Funktion 52h ( undokumentiert ) kann die Adresse vom DIB im Registerpaar ES:BX ermittelt werden. Der Pointer ( 4 Byte ) des ersten MCB ist an der Adresse ES: [ BX-4 ] zu finden

Der **Umgebungsblock ( Environment )** enthält Strings, die durch die DOS-Befehle **SET** und **PATH** definiert werden ( ASCII-String, durch Null-Zeichen beendet )

Mit dem DOS-Befehl **mem /d /p** läßt sich die aktuelle Speicherbelegung darstellen.

|                                                    |       |       |  |
|----------------------------------------------------|-------|-------|--|
| freier Speicher ( durch letzten MCB<br>verwaltet ) |       |       |  |
| Z                                                  | 00 00 | Länge |  |
| zusätzlicher Speicher<br>( von PROG1 angefordert ) |       |       |  |
| M                                                  | 00 00 | Länge |  |
| PROG1.EXE                                          |       |       |  |
| M                                                  | PSP2  | Länge |  |
| TEST1.COM                                          |       |       |  |
| M                                                  | PSP1  | Länge |  |
| DOS und BIOS                                       |       |       |  |

Ende des freien RAM-Bereiches (TPA)



00000 h  
Verwaltung des RAM-Bereiches mit MCBs

## Identifikation eines PSP-Blocks im Speicher

## Multiplex-Interrupt 2Fh

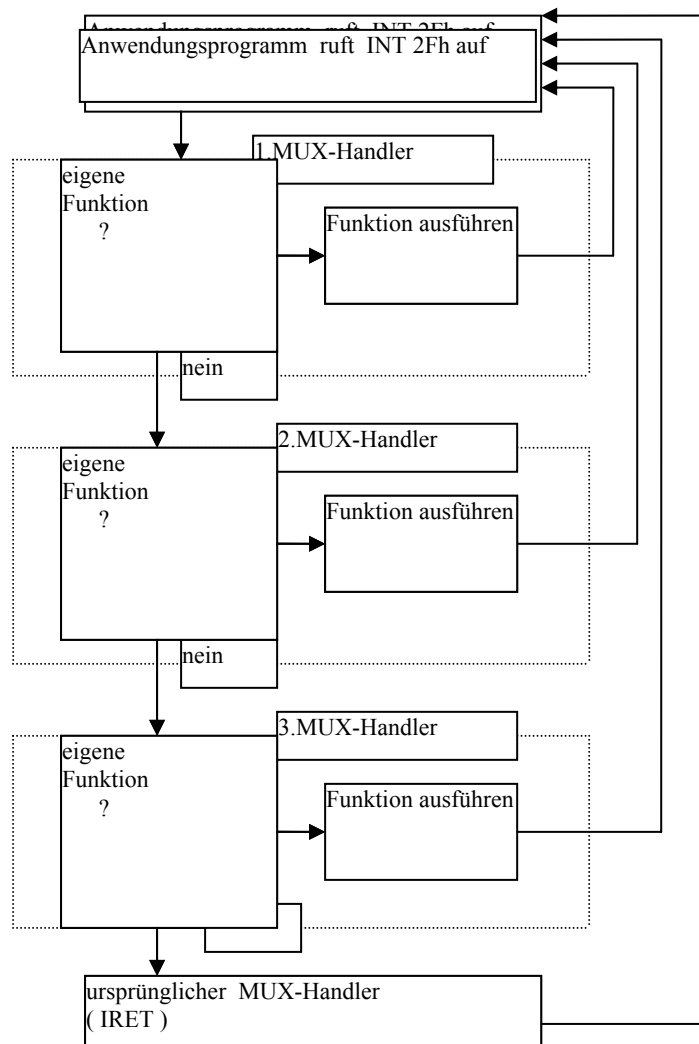
dient zur Festlegung einer Schnittstelle zwischen Prozessen. Wird für die Verwaltung von DOS-Befehlen, die sich bei ihrem Aufruf als TSR-Programme resident im Speicher installieren (PRINT, ASSIGN, SHARE, APPEND, DOSKEY, ....) verwendet und ermöglicht gewissermaßen bescheidenes Multitasking

Um nach dem resident machen mit diesem Programmen noch in Verbindung treten zu können, klinken sich diese in den **Multiplexer-Interrupt 2F h** ein, der damit für alle TSR-Programme eine Kommunikationsschnittstelle nach außen darstellt um z.B. bestimmte Parameter in dem TSR-Programm zu verändern oder das Programm aus dem Speicher zu entfernen.

Ein Programm das den MUX verwendet, muß sich selbst zunächst eine **8Bit-Kennnummer ( MUX-Code )** geben, wobei die Nummern 00-BFh für DOS-Programme reserviert sind und der Bereich von C0-FFh für Anwender -programme genutzt werden kann.

Für die Kommunikation mit dem TSR-Programm muß dieses einen eigenen Interrupt-Handler für den MUX 2Fh installieren. Bei einem Aufruf muß diese 2Fh-ISR als erstes durch Vergleich des AH-Registers mit dem MUX-Code feststellen, ob das eigene oder ein anderes Programm gemeint ist.

Bei einem Treffer wird die in den anderen Registern parametrisierte Funktion ausgeführt und das Programm mit einem IRET beendet. Anderenfalls wird zur nächsten MUX-ISR in der Multiplexer-Kette verzweigt.



| Bedeutung                                              | Eingabe / Kommentar                                                                                        | Ausgabe / Kommentar                                                                      |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------|
| AH: Enthält MUX-Code<br>Aufruf : <b>INT 2Fh</b>        | AH = 00 – BFh Reserviert von DOS<br>AH = C0h – FFh Frei für Anwendungen                                    |                                                                                          |
| Rückmeldung Installationszustand<br>Installationscheck | AL = 00 AH = MUX-Code<br>Prüfung, ob Multiplexer- ISR bereits installiert bzw ob MUX-Code bereits vergeben | AL = 00 AH = MUX-Code noch nicht installiert<br>AL = MUX-Code AH = 0 bereits installiert |
| Segmentadresse der MUX-ISR                             | AL = 01 AH = MUX-Code                                                                                      | AX = Segment                                                                             |

TSR-Anwenderprogramme die den Multiplexer 2Fh nutzen sollten die beiden Unterfunktionen AL = 00 für den **Installationscheck** und AL = 01 für die Rücklieferung der **ISR-Segmentadresse** aufweisen.  
Bei 00= Installationscheck werden AH und AL vertauscht zurückgeliefert und damit dem Aufrufer signalisiert, daß eine MUX-ISR mit dem in AH gelieferten MUX-Code bereits installiert ist.  
Die zurückgelieferte Segmentadresse der Funktion 01 wird z.B. für die Deinstallation benötigt.

## DOS-Funktion für Laden und Ausführen eines Programms (EXEC) Aufruf: INT 21h

EXEC-Funktion 4Bh INT 21h zum Laden und Starten anderer Programme

( wird auch von DOS verwendet= der Kommandoprozessor COMMAND.COM führt durch den Benutzer angegebene Programme mit Hilfe der EXEC-Funktion aus und stellt das Vater-Programm dar )

Vater-Programm ruft Kind-Programm mit 4Bh auf und das Kind-Programm erbt alle Betriebsmittel des Vater-Programms und Rückgabe von Fehlercodes von Kind auf Vater und deren Abfrage mit Funktion 4Dh.

Da .COM- und EXE-Programme den gesamten RAM-Speicher belegen, muß vor dem Ausführen der Kind-Programme der restliche Speicher mit Fktn 4Ah freigemacht werden.

| Funktion                            | Eingabe / Kommentar                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Ausgabe / Kommentar                                                                                                                                                                                                                         |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Laden und Ausführen eines Programms | <b>AH = 4B h</b><br>ES:BX -> Zeiger auf Übergabeparameterblock<br>DS:DX -> Zeiger auf Dateispezifikation<br><b>AL = 00 h</b> Unterfunktion <b>Laden und Ausführen</b><br>Übergabeparameterblock:<br>W Segmentadresse Umgebungstabelle<br>DW Zeiger auf Befehlsparameter-Zeichenkette<br>(wird kopiert in neuen PSP ab Offset 80 h )<br>DW Zeiger auf ersten FCB ( bei PSP + 5Ch )<br>DW Zeiger auf zweiten FCB( bei PSP + 6Ch )<br><b>AL = 03 h</b> Unterfkt <b>Laden eines Overlays</b> (ohne PSP)<br>Übergabeparameterblock:<br>W Segmentadresse, Ladeadresse für Datei<br>W Relozierungsparameter für Speicherabbild | CY = 0 alles o.k.<br>CY = 1 AX= codierte Fehlernr.<br>01: Funktion ungültig<br>02: Datei nicht vorhanden<br>oder Pfad ungültig<br>05: Zugriff verweigert<br>08: Speicher zu klein<br>0Ah: Umgebungstabelle ungültig<br>0Bh: Format ungültig |

## Interrupt 21h, Funktion 4Bh/00h DOS

## EXEC: anderes Programm ausführen

Diese Funktion ermöglicht es einem Programm, ein anderes Programm ausführen zu lassen, um nach dessen Ausführung selbst weiter abgearbeitet zu werden. Dazu muß der Funktion neben dem Namen des auszuführenden Programms auch die Adresse eines Parameterblocks übergeben werden, der die für die Funktion wichtigen Informationen enthält.

Eingabe AH = 4Bh  
AL = 00h  
ES:BX = FAR-Zeiger auf den Parameterblock  
DS:DX = FAR-Zeiger auf den Puffer mit dem Dateinamen des Programms

Ausgabe Carry-Flag = 0: o.k.  
Carry-Flag = 1: Fehler, in diesem Fall: AX = Fehler-Code  
1: unbekannter Funktionscode  
2: Programm nicht gefunden  
3: Programm nicht gefunden  
4: zu viele Dateien geöffnet  
5: Zugriff verweigert  
8: nicht genügend Speicherbereich  
10: falscher Environment-Block  
11: falsches Format

Der Programmname muß als ASCII-String vorliegen, der durch ein Ende-Zeichen (ASCII-Code 0) abgeschlossen wird. Er darf neben einer Gerätebezeichnung eine komplette Pfadbezeichnung und einen Dateinamen, aber keine Wildcards enthalten. Fehlt die Gerätebezeichnung oder die Pfadbezeichnung, wird auf das aktuelle Gerät bzw. auf das aktuelle Verzeichnis zugegriffen.

Es können nur EXE- oder COM-Programme zur Ausführung gebracht werden. Um eine Batch-Datei auszuführen, muß der Kommandoprozessor (COMMAND.COM) mit dem Parameter /c, gefolgt vom Namen der Batch-Datei, aufgerufen werden.

Der **Parameterblock** muß folgendes Format haben:

Byte 0-1: Segmentadresse des Environment-Blocks  
Byte 2-3: Offsetadresse der Kommandoparameter  
Byte 4-5: Segmentadresse der Kommandoparameter  
Byte 6-7: Offsetadresse des ersten FCB  
Byte 8-9: Segmentadresse des ersten FCB  
Byte 10-11: Offsetadresse des zweiten FCB  
Byte 12-13: Segmentadresse des zweiten FCB

Wird als Segmentadresse des Environment-Blocks der Wert 0 übergeben, so verfügt das aufgerufene Programm über den gleichen Environment-Block wie das aufrufende Programm.

Die Kommandozeile enthält Befehle für das auszuführende Kind-Progr und wird von DOS in den PSP des Kind-Progr. ab Adresse 80h kopiert. Die Kommandoparameter müssen im Speicher in der Art gespeichert sein, daß zunächst die Anzahl der Zeichen in der Kommandozeile als Byte abgespeichert wird. Darauf folgen die einzelnen ASCII-Zeichen, die durch ein Carriage Return (ASCII-Code 13) beendet werden. Dieses Carriage Return wird allerdings bei der Anzahl der Zeichen nicht mitgezählt.

z.B. PRGPARA DB 9,'/c DIR ',13

Der erste übergebene FCB wird ab der Adresse 5Ch, der zweite ab der Adresse 6Ch in den PSP des aufgerufenen Programms kopiert. Entnimmt das aufgerufene Programm diesen beiden FCB keine Informationen, können beliebige Werte in die FCB-Felder im Parameterblock eingetragen werden.

Nach dem Aufruf dieser Funktion sind **alle Register bis auf das CS- und das IP-Register zerstört.!!!!**

Das aufgerufene Programm verfügt über alle Handles, die auch dem aufrufenden Programm zur Verfügung stehen.

### Interrupt 21h, Funktion 4Bh / 03h DOS EXEC: anderes Programm als Overlay laden

Diese Funktion ermöglicht es einem Programm, ein anderes Programm als Overlay in den Speicher zu laden, jedoch ohne daß dieses Programm automatisch ausgeführt wird, d.h. es wird die Kontrolle an das aufrufende Programm zurückgegeben.

Der **Parameterblock** muß folgendes Format haben:

Byte 0-1: **Segmentadresse**, an die das Overlay geladen wird (die Offsetadresse dazu ist 0)  
Byte 2-3: **Relokationsfaktor**

Als Relokationsfaktor sollte bei COM-Programmen der Wert 0, bei EXE-Programmen die Segmentadresse angegeben werden, an die das Programm geladen wird.

Da kein PSP mitgeladen wird beginnt deswegen bei .COM-Programmen der Code nicht ab Offset-Adresse 100h sondern ab Adresse 00h relativ zum Ladesegment. Da sich in .COM-Programmen aber bestimmte Sprünge und Datenzugriffe auf Codebeginn 100h beziehen, muß die Startadresse für den FAR CALL-Sprung angepaßt werden:

**Ladeadresse Startadresse**  
**CS<sub>Lade</sub> : 00 = (CS<sub>Lade</sub> -10h) : 100h = gleiche physikalische Adresse**

Um .EXE-Overlays problemlos ausführen zu können, sollte in den zu ladenden EXE-Programmen der erste auszuführende Befehl definiert festgelegt werden ( z.B. an den Anfang mit Offset 00h oder auch 100h möglich )

Dann können Ladesegment und Startsegment gleichbleiben.

Nach Laden des Overlays kann das geladene Programm durch einen FAR-Call aufgerufen werden und es gilt:

**.COM-Progr.: FAR CALL nach (Ladesegment-10h) : 100h**

**.EXE-Progr.: FAR CALL nach Ladesegment : Startadresse EXE-Progr.**

Der Inhalt der Register BX, CX, DX, SI, DI, BP, CS, DS, SS und ES wird durch diese Funktion nicht verändert.

### Interrupt 21h, Funktion 4Bh/05h DOS Eigene EXECs anpassen

Applikationen, die andere Programme oder Overlays unter Umgehung der DOS-Exec-Funktion einladen, müssen sich ab der DOS-Version 5.0 dieser Funktion bedienen, um Probleme beim Nachladen der Programme und Overlays zu vermeiden.

Eingabe AH = 4Bh  
AL = 05h  
DS:DX = FAR-Zeiger auf die Exec-State-Struktur

Ausgabe Carry-Flag = 0: o.k.

Carry-Flag = 1: Fehler, in diesem Fall: AX = Fehler-Code siehe Funktion 4Bh/ 00h

Der Aufruf dieser Funktion muß zwischen dem Einladen des Programms oder Overlays und dessen Ausführung angesiedelt werden. Zwischen dem Aufruf dieser Funktion und dem Start des Programms bzw. Overlays dürfen dabei weder DOS- oder BIOS-Funktionen, noch irgendwelche anderen Software-Interrupts aufgerufen werden.

In der ExecState-Struktur werden Informationen über das Overlay bzw. Programme zusammengefaßt :

| Adr. | Inhalt                                                                                                                 | Typ     |
|------|------------------------------------------------------------------------------------------------------------------------|---------|
| +00h | Reserviert, muß 0 enthalten                                                                                            | 1 WORD  |
| +02h | 1 = EXE-Programm<br>2 = Overlay                                                                                        | 1 WORD  |
| +04h | Zeiger auf einen ASCII-String mit dem Namen des Programms bzw. Overlays<br>(Pfadangaben sind in diesem String erlaubt) | 1 PTR   |
| +08h | Segmentadresse des PSP des Programms bzw. Overlays                                                                     | 1 WORD  |
| +0Ah | Einsprungspunkt in das Programm bzw. Overlay                                                                           | 1 PTR   |
| +0Eh | Programm- bzw. Overlay-Größe inkl. PSP                                                                                 | 1 DWORD |

Länge: 12h (18 Byte) Tabelle : Aufbau der **ExecState-Struktur**

### DOS-Interrupt-Routinen für Dateien Aufruf: `int 21h`

| Funktion                                                                                             | Eingabe                                                                                                                                | Rückgabe                                                                                                   |
|------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|
| AH = 3DH: alte Datei öffnen<br><br>Datei <i>nicht</i> vorhanden:<br>Cy = 1: Dateifehler              | AL = 0: nur lesen<br>AL = 1: nur schreiben<br>AL = 2: lesen und schreiben<br>DS:DX = Stringadresse für Pfad<br>nullterminierter String | AX <= <b>handle</b><br>Cy = 0: kein Fehler<br>Cy = 1: Dateifehler<br>AX = Fehlernummer                     |
| AH = 3CH: neue Datei öffnen<br>Datei wird immer <i>neu</i> erstellt<br>vorhandene Datei wird geleert | CX = Dateiattribut = 0 setzen<br><br>DS:DX = Stringadresse für Pfad                                                                    | AX <= <b>handle</b><br>Cy = 0: kein Fehler<br>Cy = 1: Dateifehler -> AX                                    |
| AH = 3FH: geöffnete Datei<br>lesen                                                                   | BX = <b>handle</b><br>CX = Anzahl zu lesender Bytes<br>DS:DX: Pufferadresse der Daten                                                  | AX = Zahl der geles. Bytes<br>Cy = 0: kein Fehler<br>Cy = 1: Dateifehler-> AX                              |
| AH = 40H: geöffnete Datei<br>schreiben                                                               | BX = <b>handle</b><br>CX = Anzahl der Bytes<br>DS:DX: Pufferadresse der Daten                                                          | AX = Zahl der geschr. Bytes<br>Cy = 0: kein Fehler<br>Cy = 1: Dateifehler                                  |
| AH = 42H geöffnete Datei<br>positionieren                                                            | BX = <b>handle</b><br>CX:DX=Anzahl der Positionen<br>AL = 0: ab BOF (Beginn)<br>AL = 1: +/- Ist-Position<br>AL = 2: ab EOF (Ende)      | DX:AX= neue Position<br>ab BOF (Beginn)<br>Cy = 0: kein Fehler<br>Cy = 1: Dateifehler<br>AX = Fehlernummer |
| AH = 3EH: geöffnete Datei<br>schliessen                                                              | BX = <b>handle</b>                                                                                                                     | Cy = 0: kein Fehler<br>Cy = 1: Dateifehler-> AX                                                            |
| AH = 41H: Löschen einer Datei aus<br>einem Verzeichnis                                               | DS:DX = Adresse des Pfadnamens                                                                                                         | Cy = 0: kein Fehler<br>Cy = 1: Dateifehler-> AX                                                            |
| AH = 43H: Bestimmen / Setzen<br>Dateiattribut                                                        | DS:DX = Adresse des Pfadnamens<br>AL=0 Bestimme Attribute<br>AL=1 Setze Attribute ( CX ) B3=0                                          | Cy = 0: kein Fehler dann<br>CX= aktuelles Attribut<br>Cy = 1: Dateifehler dann<br>AX = Fehlernummer        |
| AH=56H: Umbenennen einer Datei                                                                       | DS:DX = Adresse alter Pfadnamen<br>ES:DI = Adresse neuer Pfadnamen                                                                     | Cy = 0: kein Fehler<br>Cy = 1: Fehler -> AX=Fehlernr                                                       |

Aufbau des Dateiattributes in CX:

|  |    |    |    |
|--|----|----|----|
|  | B2 | B1 | B0 |
|--|----|----|----|

1: Nur-Lese-Datei

1: versteckte Datei

1: Systemdatei

|                  |                                         |
|------------------|-----------------------------------------|
| CON              | Tastatur und Bildschirm                 |
| AUX              | Serielle Schnittstelle                  |
| COM1, COM2       | Serielle Schnittstellen                 |
| PRN              | Parallele Schnittstelle ( Drucker )     |
| LPT1, LPT2, LPT3 | Parallele Schnittstellen ( Drucker )    |
| NUL              | Imaginäres Gerät, das Daten verschluckt |

**Die zwingend vorgegebenen Namen der  
Ein- und Ausgabegeräte**

- 0 Standard-Eingabegerät ( CON )
- 1 Standard-Ausgabegerät ( CON )
- 2 Standardgerät zur Ausgabe von Fehlermeldungen ( CON )
- 3 Serielle Schnittstelle ( AUX )
- 4 Standard-Drucker ( PRN )

**Die Standard-Handles**

|     | HEX  | 0    | 1    | 2    | 3    | 4    | 5    | 6    | 7    | höherwertiges Halbbyte |
|-----|------|------|------|------|------|------|------|------|------|------------------------|
| HEX | BIN  | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |                        |
| 0   | 0000 | NUL  | DLE  | SP   | 0    | @    | P    | `    | p    |                        |
| 1   | 0001 | SOH  | DC1  | !    | 1    | A    | Q    | a    | q    |                        |
| 2   | 0010 | STX  | DC2  | "    | 2    | B    | R    | b    | r    |                        |
| 3   | 0011 | ETX  | DC3  | #    | 3    | C    | S    | c    | s    |                        |
| 4   | 0100 | EOT  | DC4  | \$   | 4    | D    | T    | d    | t    |                        |
| 5   | 0101 | ENQ  | NAK  | %    | 5    | E    | U    | e    | u    |                        |
| 6   | 0110 | ACK  | SYN  | &    | 6    | F    | V    | f    | v    |                        |
| 7   | 0111 | BEL  | ETB  | '    | 7    | G    | W    | g    | w    |                        |
| 8   | 1000 | BS   | CAN  | (    | 8    | H    | X    | h    | x    |                        |
| 9   | 1001 | HT   | EM   | )    | 9    | I    | Y    | i    | y    |                        |
| A   | 1010 | LF   | SUB  | *    | :    | J    | Z    | j    | z    |                        |
| B   | 1011 | VT   | ESC  | +    | ;    | K    | [    | k    | {    |                        |
| C   | 1100 | FF   | FS   | ,    | <    | L    | \    | l    |      |                        |
| D   | 1101 | CR   | GS   | -    | =    | M    | ]    | m    | }    |                        |
| E   | 1110 | SO   | RS   | .    | >    | N    | ^    | n    | ~    |                        |
| F   | 1111 | SI   | US   | /    | ?    | O    | _    | o    | DEL  |                        |

niederwertiges Halbbyte

**ASCII-Code**, 7-Bit-Code nach DIN 66003  
(Beispiel: 6 => 0011 0110<sub>2</sub> = 36 h)

| Hex-Ziffer<br>1.<br>2. | 0- | 1- | 2- | 3- | 4- | 5- | 6- | 7- | 8- | 9- | A- | B- | C- | D- | E- | F- |
|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| -0                     |    | ▶  |    | 0  | @  | P  | `  | p  | Ç  | Ê  | á  | ▤  | └  | ⊥  | α  | ≡  |
| -1                     | ☺  | ◀  | !  | 1  | A  | Q  | a  | q  | ü  | æ  | í  | ▥  | ┌  | ⊢  | β  | ±  |
| -2                     | ☹  | ↕  | "  | 2  | B  | R  | b  | r  | é  | Æ  | ó  | ▧  | ┐  | ⊣  | Γ  | ≥  |
| -3                     | ♥  | !! | #  | 3  | C  | S  | c  | s  | â  | ô  | ú  | ▨  | └  | ⊥  | π  | ≤  |
| -4                     | ♦  | ¶  | \$ | 4  | D  | T  | d  | t  | ä  | ö  | ñ  | ▩  | —  | ⊥  | Σ  | f  |
| -5                     | ♣  | §  | %  | 5  | E  | U  | e  | u  | à  | ò  | Ñ  | ▪  | +  | ⊢  | σ  | J  |
| -6                     | ♠  | —  | &  | 6  | F  | V  | f  | v  | á  | û  | ª  | ▫  | ⊢  | ⊢  | μ  | ÷  |
| -7                     | •  | ↕  | '  | 7  | G  | W  | g  | w  | ç  | ù  | º  | ▬  | └  | ⊢  | τ  | ≈  |
| -8                     | ■  | ↑  | (  | 8  | H  | X  | h  | x  | ê  | ÿ  | ¿  | ▭  | └  | ⊢  | Φ  | °  |
| -9                     | ○  | ↓  | )  | 9  | I  | Y  | i  | y  | ë  | Ö  | ⌈  | ▮  | └  | ⊢  | ⊙  | •  |
| -A                     | ◉  | →  | *  | :  | J  | Z  | j  | z  | è  | Ü  | ⌋  | ▯  | └  | ⊢  | Ω  | •  |
| -B                     | ♂  | ←  | +  | ;  | K  | [  | k  | {  | ï  | ℥  | ½  | ▰  | └  | ⊢  | δ  | ✓  |
| -C                     | ♀  | └  | <  | <  | L  | \  | l  |    | î  | £  | ¼  | ▱  | └  | ⊢  | ∞  | n. |
| -D                     | 🎵  | ↔  | -  | =  | M  | ]  | m  | }  | ï  | ¥  | ı  | ▲  | └  | ⊢  | ∅  | 2  |
| -E                     | 🎶  | ▲  | .  | >  | N  | ^  | n  | ~  | Ä  | Pt | «  | △  | └  | ⊢  | ε  | ■  |
| -F                     | ☼  | ▼  | /  | ?  | O  | _  | o  | △  | Å  | f  | »  | ▴  | └  | ⊢  | ∩  |    |

Erweiterter IBM-PC Zeichensatz für Bildschirmausgabe

| Dezimal-<br>Wert | Hex-<br>Code | ASCII-<br>Zeichen | Bedeutung                 |                                   |
|------------------|--------------|-------------------|---------------------------|-----------------------------------|
|                  |              |                   | englisch                  | deutsch                           |
| 00               | 00           | NUL               | Null                      | Füllzeichen                       |
| 01               | 01           | S0H               | Start of Heading          | Anfang des Kopfes                 |
| 02               | 02           | STX               | Start of Text             | Anfang des Textes                 |
| <b>03</b>        | <b>03</b>    | <b>ETX</b>        | <b>End of Text</b>        | <b>Ende des Textes</b>            |
| 04               | 04           | EOT               | End of Transmission       | Ende der Übertragung              |
| 05               | 05           | ENQ               | Enquiry                   | Stationsaufforderung              |
| <b>06</b>        | <b>06</b>    | <b>ACK</b>        | <b>Acknowledge</b>        | <b>Positive Rückmeldung</b>       |
| 07               | 07           | BEL               | Bell                      | Klingel                           |
| <b>08</b>        | <b>08</b>    | <b>BS</b>         | <b>Backspace</b>          | <b>Rückwärtsschritt</b>           |
| 09               | 09           | HT                | Horizontal Tabulation     | Horizontal-Tabulator              |
| <b>10</b>        | <b>0A</b>    | <b>LF</b>         | <b>Line Feed</b>          | <b>Zeilenvorschub</b>             |
| 11               | 0B           | VT                | Vertical Tabulation       | Vertikal-Tabulator                |
| 12               | 0C           | FF                | Form Feed                 | Formularvorschub                  |
| <b>13</b>        | <b>0D</b>    | <b>CR</b>         | <b>Carriage Return</b>    | <b>Wagenrücklauf</b>              |
| 14               | 0E           | SO                | Shift Out                 | Dauerumschaltung                  |
| 15               | 0F           | SI                | Shift In                  | Rückschaltung                     |
| 16               | 10           | DLE               | Data Link Escape          | Datenübertragung-Umschaltung      |
| 17               | 11           | DC1               | Device Control 1          | Gerätesteuerung 1 ( <b>XON</b> )  |
| 18               | 12           | DC2               | Device Control 2          | Gerätesteuerung 2                 |
| 19               | 13           | DC3               | Device Control 3          | Gerätesteuerung 3 ( <b>XOFF</b> ) |
| 20               | 14           | DC4               | Device Control 4          | Gerätesteuerung 4                 |
| 21               | 15           | NAK               | Negative Acknowledge      | Negative Rückmeldung              |
| 22               | 16           | SYN               | Synchronous Idle          | Synchronisierung                  |
| 23               | 17           | ETB               | End of Transmission Block | Ende des Übertragungs-Blocks      |
| 24               | 18           | CAN               | Cancel                    | Ungültig machen                   |
| 25               | 19           | EM                | End of Medium             | Ende der Aufzeichnung             |
| 26               | 1A           | SUB               | Substitute                | Substitution                      |
| <b>27</b>        | <b>1B</b>    | <b>ESC</b>        | <b>Escape</b>             | <b>Umschaltung</b>                |
| 28               | 1C           | FS                | File Separator            | Hauptgruppen-Trennung             |
| 29               | 1D           | GS                | Group Separator           | Gruppen-Trennung                  |
| 30               | 1E           | RS                | Record Separator          | Untergruppen-Trennung             |
| 31               | 1F           | US                | Unit Separator            | Teilgruppen-Trennung              |
| <b>32</b>        | <b>20</b>    | <b>SP</b>         | <b>Space</b>              | <b>Zwischenraum, Leerschritt</b>  |
| 127              | 7F           | DEL               | Delete                    | Löschen                           |

**Bedeutung der Sonderzeichen im ASCII-Code nach DIN 66003**

### Tastencodes (deutsche Tastaturbelegung)

| Taste                 | Scancode |     | ASCII/erweitert |     |       | ASCII/erweitert mit Umschalt |     |       | ASCII/erweitert mit Strg |     |       | ASCII/erweitert mit Alt |     |       |
|-----------------------|----------|-----|-----------------|-----|-------|------------------------------|-----|-------|--------------------------|-----|-------|-------------------------|-----|-------|
|                       | Dez      | Hex | Dez             | Hex | Zeich | Dez                          | Hex | Zeich | Dez                      | Hex | Zeich | Dez                     | Hex | Zeich |
| ESC                   | 1        | 01  | 27              | 1b  |       | 27                           | 1b  |       | 27                       | 1b  |       | 1                       | 01  | NUL   |
| 1 !                   | 2        | 02  | 49              | 31  | !     | 33                           | 21  | !     |                          |     |       | 120                     | 78  | NUL   |
| 2 "                   | 3        | 03  | 50              | 32  | "     | 34                           | 22  | "     | 3                        | 03  | NUL   | 121                     | 79  | NUL   |
| 3 \$                  | 4        | 04  | 51              | 33  | \$    | 21                           | 15  | °     |                          |     |       | 122                     | 7a  | NUL   |
| 4 %                   | 5        | 05  | 52              | 34  | %     | 36                           | 24  | \$    |                          |     |       | 123                     | 7b  | NUL   |
| 5 %                   | 6        | 06  | 53              | 35  | %     | 37                           | 25  | %     |                          |     |       | 124                     | 7c  | NUL   |
| 6 &                   | 7        | 07  | 54              | 36  | &     | 38                           | 26  | &     |                          |     |       | 125                     | 7d  | NUL   |
| 7 /                   | 8        | 08  | 55              | 37  | /     | 47                           | 2f  | /     |                          |     |       | 126                     | 7e  | NUL   |
| 8 (                   | 9        | 09  | 56              | 38  | (     | 40                           | 28  | (     |                          |     |       | 127                     | 7f  | NUL   |
| 9 )                   | 10       | 0a  | 57              | 39  | )     | 41                           | 29  | )     |                          |     |       | 128                     | 80  | NUL   |
| 0 =                   | 11       | 0b  | 48              | 30  | =     | 61                           | 3d  | =     |                          |     |       | 129                     | 81  | NUL   |
| ß ?                   | 12       | 0c  | 225             | e1  | ■     | 63                           | 3f  | ?     | 28                       | 1c  |       | 130                     | 82  | NUL   |
| ' `                   | 13       | 0d  | 39              | 27  | '     | 96                           | 60  | '     |                          |     |       | 131                     | 83  | NUL   |
| BKSP                  | 14       | 0e  | 8               | 08  |       | 8                            | 08  |       | 127                      | 7f  |       | 14                      | 0e  | NUL   |
| TAB                   | 15       | 0f  | 9               | 09  |       | 15                           | 0f  |       | 148                      | 94  | NUL   | 165                     | a0  | NUL   |
| Q                     | 16       | 10  | 113             | 71  | q     | 81                           | 51  | Q     | 17                       | 11  | ~Q    | 16                      | 10  | NUL   |
| W                     | 17       | 11  | 119             | 77  | w     | 87                           | 57  | W     | 23                       | 17  | ~W    | 17                      | 11  | NUL   |
| E                     | 18       | 12  | 101             | 65  | e     | 69                           | 45  | E     | 5                        | 05  | ~E    | 18                      | 12  | NUL   |
| R                     | 19       | 13  | 114             | 72  | r     | 82                           | 52  | R     | 18                       | 12  | ~R    | 19                      | 13  | NUL   |
| T                     | 20       | 14  | 116             | 74  | t     | 84                           | 54  | T     | 20                       | 14  | ~T    | 20                      | 14  | NUL   |
| Z                     | 21       | 15  | 122             | 7a  | z     | 90                           | 5a  | Z     | 26                       | 1a  | ~Z    | 44                      | 2c  | NUL   |
| U                     | 22       | 16  | 117             | 75  | u     | 85                           | 55  | U     | 21                       | 15  | ~U    | 22                      | 16  | NUL   |
| I                     | 23       | 17  | 105             | 69  | i     | 73                           | 49  | I     | 9                        | 09  | ~I    | 23                      | 17  | NUL   |
| O                     | 24       | 18  | 111             | 6f  | o     | 79                           | 4f  | O     | 15                       | 0f  | ~O    | 24                      | 18  | NUL   |
| P                     | 25       | 19  | 112             | 70  | p     | 80                           | 50  | P     | 16                       | 10  | ~P    | 25                      | 19  | NUL   |
| Ü                     | 26       | 1a  | 129             | 81  | ü     | 154                          | 9a  | ■     | 27                       | 1b  |       | 26                      | 1a  | NUL   |
| + *                   | 27       | 1b  | 43              | 2b  | +     | 42                           | 2a  | *     | 29                       | 1d  |       | 27                      | 1b  | NUL   |
| ENTER                 | 28       | 1c  | 13              | 0d  |       | 13                           | 0d  |       | 10                       | 0a  |       | 28                      | 1c  | NUL   |
| ENTER <sup>2)</sup>   | 28       | 1c  | 13              | 0d  |       | 13                           | 0d  |       | 10                       | 0a  |       | 28                      | 1c  | ERW   |
| Strg li               | 29       | 1d  |                 |     |       |                              |     |       |                          |     |       |                         |     |       |
| Strg re <sup>1)</sup> | 29       | 1d  |                 |     |       |                              |     |       |                          |     |       |                         |     |       |
| A                     | 30       | 1e  | 97              | 61  | a     | 65                           | 41  | A     | 1                        | 01  | ~A    | 30                      | 1e  | NUL   |
| S                     | 31       | 1f  | 115             | 73  | s     | 83                           | 53  | S     | 19                       | 13  | ~S    | 31                      | 1f  | NUL   |
| D                     | 32       | 20  | 100             | 64  | d     | 68                           | 44  | D     | 4                        | 04  | ~D    | 32                      | 20  | NUL   |
| F                     | 33       | 21  | 102             | 66  | f     | 70                           | 46  | F     | 6                        | 06  | ~F    | 33                      | 21  | NUL   |
| G                     | 34       | 22  | 103             | 67  | g     | 71                           | 47  | G     | 7                        | 07  | ~G    | 34                      | 22  | NUL   |
| H                     | 35       | 23  | 104             | 68  | h     | 72                           | 48  | H     | 8                        | 08  | ~H    | 35                      | 23  | NUL   |
| J                     | 36       | 24  | 106             | 6a  | j     | 74                           | 4a  | J     | 10                       | 0a  | ~J    | 36                      | 24  | NUL   |
| K                     | 37       | 25  | 107             | 6b  | k     | 75                           | 4b  | K     | 11                       | 0b  | ~K    | 37                      | 25  | NUL   |
| L                     | 38       | 26  | 108             | 6c  | l     | 76                           | 4c  | L     | 12                       | 0c  | ~L    | 38                      | 26  | NUL   |
| Ö                     | 39       | 27  | 148             | 94  | ö     | 153                          | 99  | Œ     |                          |     |       | 39                      | 27  | NUL   |
| Ä                     | 40       | 28  | 132             | 84  | ä     | 142                          | 8e  | —     |                          |     |       | 40                      | 28  | NUL   |
| ^ °                   | 41       | 29  | 94              | 5e  | ^     | 248                          | f8  | ⋮     |                          |     |       | 41                      | 29  | NUL   |
| Umsch li              | 42       | 2a  |                 |     |       |                              |     |       |                          |     |       |                         |     |       |
| # '                   | 43       | 2b  | 35              | 23  | #     | 39                           | 27  | '     |                          |     |       | 43                      | 2b  | NUL   |
| Y                     | 44       | 2c  | 121             | 79  | y     | 89                           | 59  | Y     | 25                       | 19  | ~Y    | 21                      | 15  | NUL   |
| x                     | 45       | 2d  | 120             | 78  | x     | 88                           | 58  | X     | 24                       | 18  | ~X    | 45                      | 2d  | NUL   |

|                         |    |    |     |    |     |          |    |     |     |    |     |          |    |     |
|-------------------------|----|----|-----|----|-----|----------|----|-----|-----|----|-----|----------|----|-----|
| C                       | 46 | 2e | 99  | 63 | c   | 67       | 43 | C   | 3   | 03 | ~C  | 46       | 2e | NUL |
| V                       | 47 | 2f | 118 | 76 | v   | 86       | 56 | V   | 22  | 16 | ~V  | 47       | 2f | NUL |
| B                       | 48 | 30 | 98  | 62 | b   | 66       | 42 | B   | 2   | 02 | ~B  | 48       | 30 | NUL |
| N                       | 49 | 31 | 110 | 6e | n   | 78       | 4e | N   | 14  | 0e | ~N  | 49       | 31 | NUL |
| M                       | 50 | 32 | 109 | 6d | m   | 77       | 4d | M   | 13  | 0d | ~M  | 50       | 32 | NUL |
| , ;                     | 51 | 33 | 44  | 2c | ,   | 59       | 3b | ;   |     |    |     | 51       | 33 | NUL |
| . :                     | 52 | 34 | 46  | 2e | .   | 58       | 3a | :   |     |    |     | 52       | 34 | NUL |
| - _                     | 53 | 35 | 45  | 2d | -   | 95       | 5f | _   | 31  | 1f |     | 53       | 35 | NUL |
| +2)                     | 53 | 35 | 47  | 2f | /   | 47       | 2f | /   | 149 | 95 | ERW | 164      | a4 | ERW |
| Umsch re                | 54 | 36 |     |    |     |          |    |     |     |    |     |          |    |     |
| Druck                   | 55 | 37 | 42  | 2a | *   | INT 5h4) |    |     |     |    |     | SysReq5) |    |     |
| x <sup>2)</sup>         | 55 | 37 | 42  | 2a | *   | 42       | 2a | *   | 150 | 96 | ERW | 55       | 37 | NUL |
| Alt li                  | 56 | 38 |     |    |     |          |    |     |     |    |     |          |    |     |
| Alt re <sup>1)</sup>    | 56 | 38 |     |    |     |          |    |     |     |    |     |          |    |     |
| leer                    | 57 | 39 | 32  | 20 | SPC | 32       | 20 | SPC | 32  | 20 | SPC | 32       | 20 | SPC |
| CAPS-Lock               | 58 | 3a |     |    |     |          |    |     |     |    |     |          |    |     |
| F1                      | 59 | 3b | 59  | 3b | NUL | 84       | 54 | NUL | 94  | 5e | NUL | 104      | 5e | NUL |
| F2                      | 60 | 3c | 60  | 3c | NUL | 85       | 55 | NUL | 95  | 5f | NUL | 105      | 5f | NUL |
| F3                      | 61 | 3d | 61  | 3d | NUL | 86       | 56 | NUL | 96  | 60 | NUL | 106      | 60 | NUL |
| F4                      | 62 | 3e | 62  | 3e | NUL | 87       | 57 | NUL | 97  | 61 | NUL | 107      | 61 | NUL |
| F5                      | 63 | 3f | 63  | 3f | NUL | 88       | 58 | NUL | 98  | 62 | NUL | 108      | 62 | NUL |
| F6                      | 64 | 40 | 64  | 40 | NUL | 89       | 59 | NUL | 99  | 63 | NUL | 109      | 63 | NUL |
| F7                      | 65 | 41 | 65  | 41 | NUL | 90       | 5a | NUL | 100 | 64 | NUL | 110      | 64 | NUL |
| F8                      | 66 | 42 | 66  | 42 | NUL | 91       | 5b | NUL | 101 | 65 | NUL | 111      | 65 | NUL |
| F9                      | 67 | 43 | 67  | 43 | NUL | 92       | 5c | NUL | 102 | 66 | NUL | 112      | 66 | NUL |
| F10                     | 68 | 44 | 68  | 44 | NUL | 93       | 5d | NUL | 103 | 67 | NUL | 113      | 67 | NUL |
| NUM-Lock                | 69 | 45 |     |    |     |          |    |     |     |    |     |          |    |     |
| NUM-Lock <sup>6)</sup>  | 69 | 45 |     |    |     |          |    |     |     |    |     |          |    |     |
| Pause <sup>7)</sup>     | 69 | 45 |     |    |     |          |    |     |     |    |     |          |    |     |
| Scroll <sup>6)</sup>    | 70 | 46 |     |    |     |          |    |     |     |    |     |          |    |     |
| POS 1                   | 71 | 47 | 71  | 47 | NUL | 55       | 37 | 7   | 119 | 77 | NUL |          |    |     |
| POS 1 <sup>3)</sup>     | 71 | 47 | 71  | 47 | ERW | 71       | 47 | ERW | 119 | 77 | ERW | 151      | 97 | ERW |
| Cursor ob               | 72 | 48 | 72  | 48 | NUL | 56       | 38 | 8   |     |    |     |          |    |     |
| Cursor ob <sup>3)</sup> | 72 | 48 | 72  | 48 | ERW | 72       | 48 | ERW | 141 | 8d | ERW | 152      | 98 | ERW |
| Bild ob                 | 73 | 49 | 73  | 49 | NUL | 57       | 39 | 9   | 132 | 84 | NUL |          |    |     |
| Bild ob <sup>3)</sup>   | 73 | 49 | 73  | 49 | ERW | 73       | 49 | ERW | 132 | 84 | ERW | 153      | 99 | ERW |
| _8)                     | 74 | 4a | 45  | 2d | -   | 45       | 2d | -   |     |    |     |          |    |     |
| Cursor li               | 75 | 4b | 75  | 4b | NUL | 52       | 34 | 4   | 115 | 73 | NUL |          |    |     |
| Cursor li <sup>3)</sup> | 75 | 4b | 75  | 4b | ERW | 75       | 4b | ERW | 115 | 73 | ERW | 155      | 9b | ERW |
| 58)                     | 76 | 4c |     |    |     | 53       | 35 | 5   | 143 | 8f | NUL |          |    |     |
| Cursor re               | 77 | 4d | 77  | 4d | NUL | 54       | 36 | 6   | 116 | 74 | NUL |          |    |     |
| Cursor re <sup>3)</sup> | 77 | 4d | 77  | 4d | ERW | 77       | 4d | ERW | 116 | 74 | ERW | 157      | 9d | ERW |
| +8)                     | 78 | 4e | 43  | 2b | +   | 43       | 2b | +   | 144 | 90 | NUL | 78       | 4e | NUL |
| Ende                    | 79 | 4f | 79  | 4f | NUL | 49       | 31 | 1   | 117 | 75 | NUL |          |    |     |
| Ende <sup>3)</sup>      | 79 | 4f | 79  | 4f | ERW | 79       | 4f | ERW | 117 | 75 | ERW | 159      | 9f | ERW |
| Cursor un               | 80 | 50 | 80  | 50 | NUL | 50       | 32 | 2   | 145 | 91 | NUL |          |    |     |
| Cursor un <sup>3)</sup> | 80 | 50 | 80  | 50 | ERW | 80       | 50 | ERW | 145 | 91 | ERW | 160      | a0 | ERW |
| Bild un                 | 81 | 51 | 81  | 51 | NUL | 51       | 33 | 3   | 118 | 76 | NUL |          |    |     |
| Bild un <sup>3)</sup>   | 81 | 51 | 81  | 51 | ERW | 81       | 51 | ERW | 118 | 76 | ERW | 161      | a1 | ERW |
| Einf <sup>3)</sup>      | 82 | 52 | 82  | 52 | NUL | 48       | 30 | 0   | 146 | 92 | NUL |          |    |     |
| Einf <sup>3)</sup>      | 82 | 52 | 82  | 52 | ERW | 82       | 52 | ERW | 146 | 92 | ERW | 162      | a2 | ERW |
| Entf <sup>3)</sup>      | 83 | 53 | 83  | 53 | NUL | 44       | 2c | ,   | 147 | 93 | NUL |          |    |     |
| Entf                    | 83 | 53 | 83  | 53 | ERW | 83       | 53 | ERW | 147 | 93 | ERW | 163      | a3 | ERW |
| < > <sup>1)</sup>       | 86 | 56 | 60  | 3c | <   | 62       | 62 | >   |     |    |     |          |    |     |
| F11                     | 87 | 57 | 133 | 85 | ERW | 135      | 87 | ERW | 137 | 89 | ERW | 139      | 8b | ERW |
| F12                     | 88 | 58 | 134 | 86 | ERW | 136      | 88 | ERW | 138 | 8a | ERW | 140      | 8c | ERW |

1) alphanumerischer Block; nur MF II mit Vorsatzbyte 0eh

2) numerischer Block; nur MF II mit Vorsatzbyte 0eh

3) im abgesetzten Steuerblock bei MF II

4) Bildschirmausdruck über INT 05h unter DOS

5) SysReq entsprechend INT 15h, Funktion 85h

6) MF II mit Vorsatzbyte e0h

7) nur MF II mit Vorsatzbyte e1h

8) numerischer Block

Tastencodes (deutsche Tastaturbelegung)

## Die parallele Schnittstelle PIO 8255 ( MVUS 80535 )

Nach einem Reset sind alle Ports als Eingang geschaltet; die Datenregister sind gelöscht.

A-Port Ausgabe: LED      Eingabe: Kippschalter  
B-Port Ausgabe: LED      Eingabe: Kippschalter  
C-Port Ausgabe: LED      Eingabe: Kippschalter und Taster  
Steuerport nur beschreibbar, nicht rücklesbar

**Steuerbyte** der Betriebsart 0 (einfache Ein-/Ausgabe):

```

1  0  0  x  x  0  x  x
                                0: Cl-Port = aus
                                1: Cl-Port = ein
                                0: B-Port = aus
                                1: B-Port = ein
                                0: Ch-Port = aus
                                1: Ch-Port = ein
                                0: A-Port = aus
                                1: A-Port = ein

```

| A-Port  | B-Port  | Ch-Port | Cl-Port | Steuerbyte |
|---------|---------|---------|---------|------------|
| Ausgang | Ausgang | Ausgang | Ausgang | 80H        |
| Ausgang | Ausgang | Ausgang | Eingang | 81H        |
| Ausgang | Ausgang | Eingang | Ausgang | 88H        |
| Ausgang | Ausgang | Eingang | Eingang | 89H        |
| Ausgang | Eingang | Ausgang | Ausgang | 82H        |
| Ausgang | Eingang | Ausgang | Eingang | 83H        |
| Ausgang | Eingang | Eingang | Ausgang | 8AH        |
| Ausgang | Eingang | Eingang | Eingang | 8BH        |
| Eingang | Ausgang | Ausgang | Ausgang | 90H        |
| Eingang | Ausgang | Ausgang | Eingang | 91H        |
| Eingang | Ausgang | Eingang | Ausgang | 98H        |
| Eingang | Ausgang | Eingang | Eingang | 99H        |
| Eingang | Eingang | Ausgang | Ausgang | 92H        |
| Eingang | Eingang | Ausgang | Eingang | 93H        |
| Eingang | Eingang | Eingang | Ausgang | 9AH        |
| Eingang | Eingang | Eingang | Eingang | 9BH        |

## Der Interruptsteuerbaustein PIC 8259 im PC

| I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 |
|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|

**Port 21 h Master** Maskenregister ( Interruptfreigabe ) 0 = frei 1 = gesperrt

**Port A1 h Slave**

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | x | x | x |
|---|---|---|---|---|---|---|---|

**Port 20 h Master** Bestätigungsregister

IRQ0: xxx = 000 bis IRQ7: xxx = 111

**Port A0 h Slave**

Die beiden Interruptsteuerbaustein PIC 8259 A ( Master und Slave ) werden beim Neustart durch das Betriebssystem durch 4 ICWS ( Initialisierungsworte ) initialisiert auf:

- steigende Flanke **Flankentriggerung** *mit Bit LTIM = 0 im ICW1*
- zwei kaskadierte PICs **Master-Slave** *mit Bit SNGL = 0 im ICW1*
- 8086-Modus -> Controller legt **Vektor-Nummer (Byte)** auf Datenbus *mit Bit  $\mu$ PM = 1 im ICW4*  
 Master -> Vektor-Nr. 08 h ( IRQ0 ) bis 0F h ( IRQ7 ) und *mit Bits Off7-Off3 im ICW2*  
 Slave -> Vektor-Nr. 70 h ( IRQ0 ) bis 77 h ( IRQ7 )
- festliegende **hierarchische Priorität**: Eingang 0 = IRQ0 – höchste Eingang 7= IRQ7 – niedrigste
- normaler **Nestet-Modus**-> verschachtelte Interrupt-Bearbeitung möglich, d.h. ein Interrupt höherer Priorität kann eine laufende Interruptbearbeitung niedriger Priorität unterbrechen. *mit Bit SFNM = 0 im ICW4*
- **manueller EOI** ( End of Interrupt )- Befehl signalisiert der CPU eine beendete Interruptbearbeitung. *mit Bit AEOI = 0 im ICW4*
- ein **unspezifischer EOI-Befehl** setzt höchst priorisiertes ISR-Bit zurück  
 ( Schreiben EOI-Befehls (20h ) nach Port 20h/A0h , durch hierarchische Prioritäten im PC üblich )
- ein **spezifischer EOI-Befehl** (6xh ) setzt das angegebene Bit im ISR zurück

Im Betrieb muß jeder Interrupteingang durch Schreiben eines OCW1 in das Maskenregister ( Port 21h/A1h ) freigegeben ( 0 ) bzw. gesperrt ( 1 ) werden. Am Ende eines Interruptprogramms ( Interrupt-Service-Routine ) ist die Annahme im Bestätigungsregister ( Port 20h/A0h ) durch Schreiben eines OCW2 mit EOI-Befehl zu bestätigen..

### Adressen und Steuerbytes dieser Betriebsart im PC mit spezifischem EOI

| Gerät      | Interr. | Freigabe | Sperre | EOI-Best. | Vektor | Adresse |
|------------|---------|----------|--------|-----------|--------|---------|
| Timer 0    | IRQ0    | AND 0FEH | OR 01H | 60H       | 08H    | 0:0020H |
| Tastatur   | IRQ1    | AND 0FDH | OR 02H | 61H       | 09H    | 0:0024H |
| PIC2-Slave | IRQ2    | AND 0FBH | OR 04H | 62H       | 0AH    | 0:0028H |
| COM 2      | IRQ3    | AND 0F7H | OR 08H | 63H       | 0BH    | 0:002CH |
| COM 1      | IRQ4    | AND 0EFH | OR 10H | 64H       | 0CH    | 0:0030H |
| LPT2       | IRQ5    | AND 0DFH | OR 20H | 65H       | 0DH    | 0:0034H |
| Disk       | IRQ6    | AND 0BFH | OR 40H | 66H       | 0EH    | 0:0038H |
| LPT1       | IRQ7    | AND 07FH | OR 80H | 67H       | 0FH    | 0:003CH |

### Der Interruptsteuerbaustein 8259A ( Master , voreingestellt )

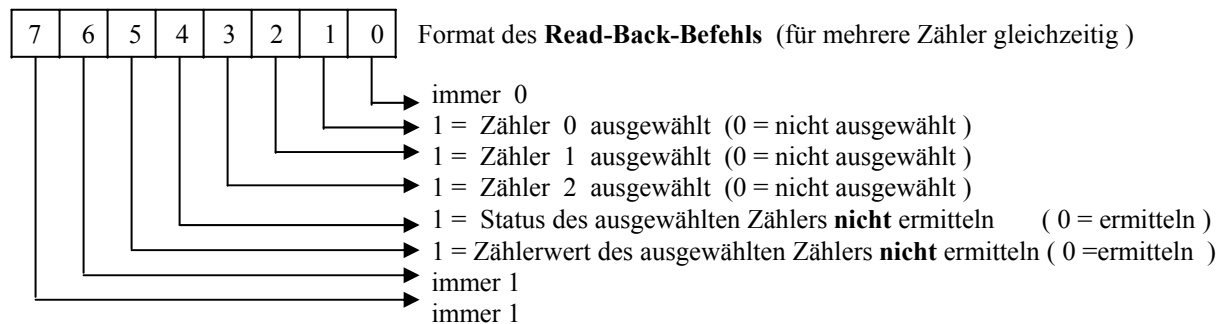
**Beispiel:** Zurücksetzen der ISR-Bit nach Bedienung des IRQ13 – Coprozessor durch unspezifischen EOI-Befehl 20 h

```
MOV AL, 20 h ; EOI-Befehl
OUT 0A0 h, AL ; an Slave
OUT 20 h, AL ; an Master
```

## Der Timerbaustein PIT 8254

Die Anschlüsse Clk, Gate und Out sind im PC mit dem Takt und der Gate-Steuerung ( +5 V) fest verbunden.  
Aufbau des Steuerbytes für den Steuerport 43h:

| SC1               | SC0 | RL1             | RL0 | M2                         | M1 | M0      | BCD |
|-------------------|-----|-----------------|-----|----------------------------|----|---------|-----|
| Timerauswahl      |     | Reihenfolge     |     | Betriebsart                |    | Zählung |     |
| 00: Timer_0       |     | 01: nur Low     |     | 000: warten (Befehl)       |    | 0: dual |     |
| 01: Timer_1       |     | 10: nur High    |     | 001: warten (Gate)         |    | 1: BCD  |     |
| 10: Timer_2       |     | 11: Low/High    |     | x10: periodische Impulse   |    |         |     |
| 11: Zähler/Status |     | 00: Zählerstand |     | x11: Frequenzteiler        |    |         |     |
| rücklesen         |     | festhalten      |     | 100: Einzelimpuls (Befehl) |    |         |     |
|                   |     |                 |     | 101: Einzelimpuls (Gate)   |    |         |     |



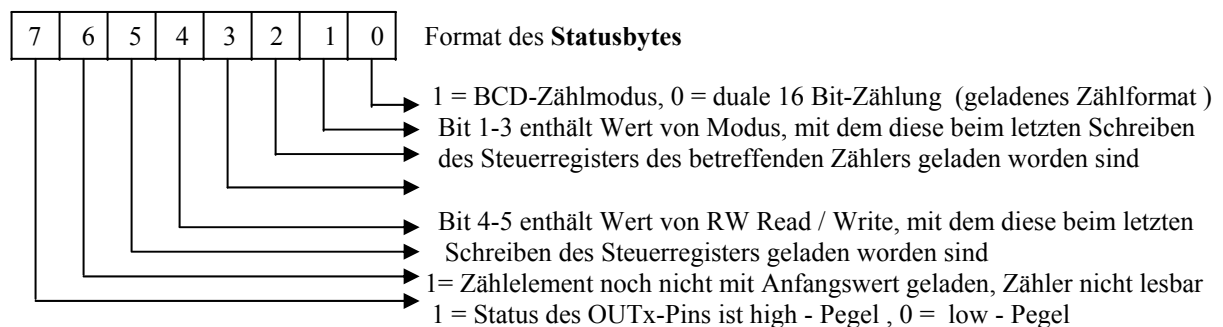
### Beispiel:

Modus des Zählers 0  
ermitteln

```

MOV AL,11100010b    ;Read-Back nur Modus
OUT 43h,AL           ;an Steuerregister
IN  AL,40h            ;Modus für Timer 0 ein

```



Im obigen **Beispiel** könnte im PC das zurückgelieferte Modusbyte wie folgt aussehen:

Statusregister: 0 0 1 1 0 1 0 0 ( INT 08h Systemuhr )

-> OUT-Pin auf 0V

-> Anfangszählwert geladen

--> Lesen / Schreiben beider Byte

---> Zählermodus 2 (per. Ratengenerator)

-> duale Zählung

## Der Uhrenbaustein RTC

| Byte | Adresse | Inhalt                        | Byte  | Adresse | Inhalt                                        |
|------|---------|-------------------------------|-------|---------|-----------------------------------------------|
| 0    | 00h     | Sekunde <sup>*)</sup>         | 22    | 16h     | Basisspeicher (höherw. Byte)                  |
| 1    | 01h     | Alarmsekunde <sup>*)</sup>    | 23    | 17h     | Extended Memory (niederwert. Byte) laut SETUP |
| 2    | 02h     | Minute <sup>*)</sup>          |       |         |                                               |
| 3    | 03h     | Alarmminute <sup>*)</sup>     | 24    | 18h     | Extended Memory (höherwert. Byte) laut SETUP  |
| 4    | 04h     | Stunde <sup>*)</sup>          |       |         |                                               |
| 5    | 05h     | Alarmstunde <sup>*)</sup>     | 25    | 19h     | Erweiterungsbyte 1. Festplatte                |
| 6    | 06h     | Tag der Woche <sup>*)</sup>   | 26    | 1ah     | Erweiterungsbyte 2. Festplatte                |
| 7    | 07h     | Tag des Monats <sup>*)</sup>  | 27-31 | 1bh-1fh | reserviert                                    |
| 8    | 08h     | Monat <sup>*)</sup>           | 32-39 | 20h-27h | Parameter Festplattentyp 48                   |
| 9    | 09h     | Jahr <sup>*)</sup>            | 40-45 | 28h-2dh | reserviert                                    |
| 10   | 0ah     | Statusregister A              | 46    | 2eh     | Prüfsumme (niederw. Byte)                     |
| 11   | 0bh     | Statusregister B              | 47    | 2fh     | Prüfsumme (höherw. Byte)                      |
| 12   | 0ch     | Statusregister C              | 48    | 30h     | Extended Memory (niederwert. Byte) laut POST  |
| 13   | 0dh     | Statusregister D              |       |         |                                               |
| 14   | 0eh     | Diagnosestatus                | 49    | 31h     | Extended Memory (höherwert. Byte) laut POST   |
| 15   | 0fh     | Shutdown-Status               |       |         |                                               |
| 16   | 10h     | Typ der Diskettenlaufwerke    | 50    | 32h     | Jahrhundert <sup>*)</sup>                     |
| 17   | 11h     | reserviert                    | 51    | 33h     | Setup-Informationen                           |
| 18   | 12h     | Typ der Festplattenlaufwerke  | 52    | 34h     | reserviert                                    |
| 19   | 13h     | reserviert                    | 53-60 | 35h-3ch | Parameter Festplattentyp 49                   |
| 20   | 14h     | Gerätebyte                    | 61-63 | 3dh-3fh | reserviert                                    |
| 21   | 15h     | Basisspeicher (niederw. Byte) |       |         |                                               |

<sup>\*)</sup> üblicherweise binärkodierte Dezimalzahlen (1 Byte)

### A - Steuerregister

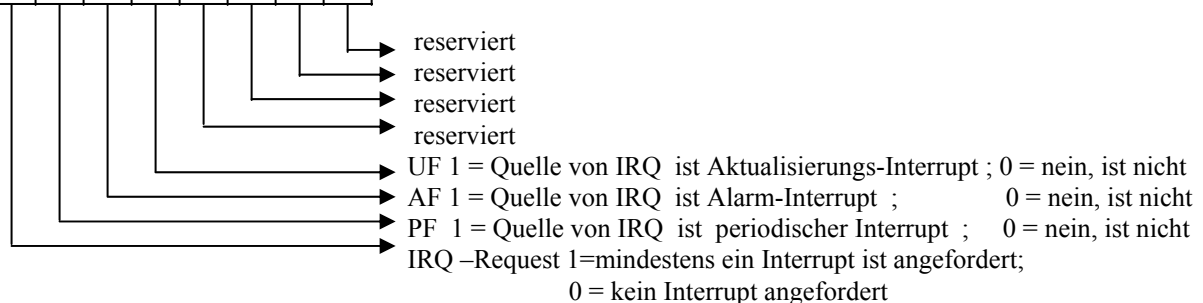
| UIP                   | DV2 | DV1 | DV0 | RS3                                  | RS2 | RS1 | RS0 |
|-----------------------|-----|-----|-----|--------------------------------------|-----|-----|-----|
| Oszillator und Teiler |     |     |     | Teiler für SQW und period. Interrupt |     |     |     |
| 0 1 0: ein            |     |     |     | 0 0 0: gesperrt                      |     |     |     |
|                       |     |     |     | 0 0 1: 3.90625 ms 256 Hz             |     |     |     |
|                       |     |     |     | 0 1 0: 7.8125 ms 128 Hz              |     |     |     |
|                       |     |     |     | 0 1 1: 122.070 us 8.192 kHz          |     |     |     |
|                       |     |     |     | 0 1 0: 244.141 us 4.096 kHz          |     |     |     |
|                       |     |     |     | 0 1 1: 488.281 us 2.048 kHz          |     |     |     |
|                       |     |     |     | 0 1 1: 976.5625 us 1.024 kHz         |     |     |     |
|                       |     |     |     | 0 1 1: 1.953125 ms 512 Hz            |     |     |     |
|                       |     |     |     | 1 0 0: 3.90625 ms 256 Hz             |     |     |     |
|                       |     |     |     | 1 0 0: 7.8125 ms 128 Hz              |     |     |     |
|                       |     |     |     | 1 0 1: 15.625 ms 64 Hz               |     |     |     |
|                       |     |     |     | 1 0 1: 31.25 ms 32 Hz                |     |     |     |
|                       |     |     |     | 1 1 0: 62.5 ms 16 Hz                 |     |     |     |
|                       |     |     |     | 1 1 0: 125 ms 8 Hz                   |     |     |     |
|                       |     |     |     | 1 1 1: 250 ms 4 Hz                   |     |     |     |
|                       |     |     |     | 1 1 1: 500 ms 2 Hz                   |     |     |     |

### B - Steuerregister

| SET | PIE | AIE | UIE | SQWE                                           | DM | 24/12 | DSE |
|-----|-----|-----|-----|------------------------------------------------|----|-------|-----|
|     |     |     |     | 0: nicht umstellen<br>1: Sommer/Winter         |    |       |     |
|     |     |     |     | 0: 24-Stunden Betrieb<br>1: 12-Stunden Betrieb |    |       |     |
|     |     |     |     | 0: BCD Daten<br>1: duale Daten                 |    |       |     |
|     |     |     |     | 0: SQW = Low<br>1: SQW = Rechteckausgang       |    |       |     |
|     |     |     |     | 1: Update - Interrupt frei                     |    |       |     |
|     |     |     |     | 1: Alarm - Interrupt frei                      |    |       |     |
|     |     |     |     | 1: Periodischer Interrupt frei                 |    |       |     |
|     |     |     |     | 0: Update jede Sekunde                         |    |       |     |
|     |     |     |     | 1: Update gesperrt                             |    |       |     |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|

### Statusregister C (Interruptanzeige, durch Lesen zurückgesetzt)



UF, AF und PF werden durch die jeweilige Aktivität gesetzt ( für Polling-Abfrage ), und dies ist nicht abhängig von der Interrupt-Freigabe im Statusregister B

## Hardware-Programmierung in C

Wichtige Funktionen und Makros in dos.h (zuordnen mit #include <dos.h>) aus [11]

| Ergebnis      | C – Funktion                                                 | Aufgabe                                        |
|---------------|--------------------------------------------------------------|------------------------------------------------|
| unsigned char | <b>inp</b> ( Port )<br><b>inportb</b> ( Port )               | liefert Byte vom Eingabeport                   |
| int           | <b>inport</b> ( Port )                                       | liest Wort (16 Bit) vom Eingabeport            |
|               | <b>outp</b> ( Port, Byte )<br><b>outportb</b> ( Port, Byte ) | schreibt Byte nach Ausgabeport                 |
|               | <b>outport</b> ( Port, Word )                                | schreibt Wort (16 Bit) nach Ausgabeport        |
| unsigned char | <b>peekb</b> (Segment, Offset )                              | liefert Speicherbyte aus Adresse Seg : Off     |
|               | <b>pokeb</b> (Segment, Offset, Byte )                        | schreibt Byte nach Speicher Seg : Off          |
| Int           | <b>peek</b> (Segment, Offset )                               | liefert Speicherwort (16 Bit) aus Seg : Off    |
|               | <b>poke</b> (Segment, Offset, Wort )                         | schreibt Wort (16 Bit) nach Speicher Seg : Off |
| unsigned int  | <b>FP_SEG</b> ( far-Zeiger )                                 | liefert Segmentanteil                          |
| unsigned int  | <b>FP_OFF</b> ( far-Zeiger )                                 | liefert Offsetanteil                           |
| far – Zeiger  | <b>MK_FP</b> ( Segment, Offset )                             | liefert Zeiger auf Adresse                     |

### Hinweise für die hardwarenahe C-Programmierung :

- Zuordnung von Systemmakros mit #include <dos.h>
- Porteingabe mit **inp**(Port), Portausgabe mit **outp**(Port, Byte)
- Speicherwort schreiben: **poke**(Segm., Offs., Wert) lesen: **peek**(Segm., Offs.)
- **keine** switch-case-Auswahl, keine Objekte und Klassen, keine 32-bit-Division für Datentyp int
- vorbesetzte Felder (Konstanten) nicht lokal, sondern global oder **static** anlegen
- Funktionen **vor** main als Prototyp deklarieren und **nach** main definieren

### Funktions-Prototyp;

**Kopfzeile;**

**main ( )**

**{ }**

**Funktions-Definitionen hinter main !! :**

| Assembler                             |                             | C-Programmierung |                                   |
|---------------------------------------|-----------------------------|------------------|-----------------------------------|
| Portausgabe und Hex-Zahlendarstellung | mov al, 0FFh<br>out 60H, AL |                  | outp ( 0x60, 0xFF );<br>Port Wert |
| NICHT                                 | not al                      | ~                | a = ~ a ;                         |
| UND                                   | and al, 0fh                 | &                | a = b & c ;                       |
| ODER                                  | or al, ah                   |                  | a = b   c ;                       |
| EODER                                 | xor ax, ax                  | ^                | a = b ^ c ;                       |
|                                       |                             |                  |                                   |
| schiebe logisch links                 | shl ax, 1                   | <<               | a = a << 1 ;                      |
| logisch rechts                        | shr ax, 1                   | >>               | a = a >> 1 ; // unsigned var      |
| arithmetisch rechts                   | sar ax, 1                   | >>               | s = s >> 1 ; // signed var        |
| rotieren links                        | rol ax, 1                   | rotl ( x, n )    | a = _rotl ( a, 1 ) ;              |
| rotieren rechts                       | ror ax, 1                   | rotr ( x, n )    | a = _rotr ( a, 1 ) ;              |

Die Rotierfunktionen sind vordefiniert in <stdlib.h>

Wichtige DOS-Betriebssystemfunktionen in dos.h ( zuordnen mit #include <dos.h> )

| Ergebnis | Funktion                         | Aufgabe                                                                                             |
|----------|----------------------------------|-----------------------------------------------------------------------------------------------------|
| ax       | <b>int86</b> ( Nr., &Ein, &Aus ) | BIOS-Interrupt mit Nummer 0...255<br>&Ein / &Aus = Zustand Prozessorregister<br>vor/nach dem Aufruf |
| ax       | <b>intdos</b> ( &Ein, &Aus )     | DOS-Interrupt 0x21                                                                                  |
| Zeiger   | <b>getvect</b> ( Nummer )        | liefert alten Vector als far – Zeiger                                                               |
|          | <b>setvect</b> (Nr., Funktion )  | setzt Interruptvektor auf Funktion                                                                  |
|          | <b>geninterrupt</b> (Nummer)     | löst Interrupt ( 0 ... 255 )                                                                        |
|          | <b>getdate</b> (&Datum)          | Datum nach vordef. Strukturtyp date<br>.da_year .da_day .da_mon                                     |
|          | <b>gettime</b> (&Uhrzeit )       | Zeit nach vordef. Strukturtyp time<br>.ti_min .hour .hund .sec                                      |
|          | <b>delay</b> (Millisekunden)     | Wartezeit in ms ( unsigned int )                                                                    |
|          | <b>sleep</b> (Sekunden )         | Wartezeit in sek ( unsigned int )                                                                   |
|          | <b>sound</b> (Frequenz )         | Lautsprecher ein (unsigned int )                                                                    |
|          | <b>nosound</b> ( )               | Lautsprecher aus                                                                                    |
|          | <b>disable</b> ( )               | INTR-Interrupt sperren                                                                              |
|          | <b>enable</b> ( )                | INTR-Interrupt freigeben                                                                            |

**Vordefinierte DOS-Betriebssystemfunktionen in < dos.h >**

Für die Prozessorregister definiert dos.h eine **union REGS** die aus den beiden Strukturen x ( Wortregister ax,... di, cflag, flags ) und h ( Byteregister al, ... dh )

**Beispiel:** Ermittlung von Datum und Uhrzeit mit DOS Interrupt 0x21

```
union REGS reg;           // REGS in dos.h vordefiniert
reg.h.ah = 0x2A ;          // ah = 0x2A: Datum auslesen
int86 ( 0x21, &reg, &reg ); // 0x21 mit BIOS-Funktion
printf ( "\n%u . %u . %u ", reg.h.dl, reg.h.dh, reg.x.cx );
reg.h.ah = 0x2C ;          // ah = 0x2C: Uhrzeit auslesen
intdos ( &reg, &reg );      // 0x21 mit DOS-Funktion
printf ( "\n%u : %u : %u ", reg.h.ch, reg.h.cl, reg.h.dh );
```

| Ergebnis | Funktion                                                                                           | Aufgabe                                                                                                                |
|----------|----------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
| int      | <b>bioscom</b> (Komm., Byte, Port )                                                                | Int 0x14: serielle Schnittstelle                                                                                       |
| int      | <b>biosprint</b> (Komm., Byte, Port )                                                              | int 0x17: Druckerschnittstelle                                                                                         |
| long int | <b>biostime</b> ( Kommando, Wert )<br><b>biostime</b> (0, 0 )                                      | Int 0x1A: Timer Tic Variable<br>liefert long int Zählerstand                                                           |
| int      | <b>bioskey</b> (Kommando )<br><b>bioskey</b> ( 0 )<br><b>bioskey</b> ( 1 )<br><b>bioskey</b> ( 2 ) | Int 0x16 Tastaturabfrage<br>liefert Scan- und ASCII-Code<br>liefert 0, wenn keine Taste<br>liefert Modifizierer-Tasten |

**Vordefinierte BIOS-Betriebssystemfunktionen in < bios.h >**

## Einbinden von Assembler in die Hochsprachen C bzw. C++

Für das direkte Ansprechen von Hardware oder auch für zeitkritische Teile werden Assembler-Routinen bzw. Sequenzen auch in höhere Programmiersprachen eingebaut.

Mit Hilfe des in Borland C++3.0 integrierten **Inline-Assemblers** (BASM) lassen sich ohne separate Assemblierung / Compilierung ganze Assembler-Sequenzen in C / C++ -Programme einbinden.

Ein Befehl in Assemblersprache wird dabei innerhalb eines C-Programms mit dem Schlüsselwort **asm** kenntlich gemacht. Eine Reihe von asm - Anweisungen müssen in geschweifter Klammer stehen

```
asm {
    pop ax ; pop ds
    iret
}
```

Des weiteren ist zu beachten, daß C zwischen Groß- und Kleinschreibung bei Bezeichnern unterscheidet.

**C-Symbole** können in **asm - Anweisungen** direkt verwendet werden.

In asm - Anweisungen können keine Labels definiert werden und deswegen müssen Assembler-Sprungbefehle zur Kennzeichnung der Sprungziele goto - Labels von C verwenden.

Beispiele:

|                                                                                                                   |                                                                                           |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|
| <pre>myfunc() {     int i ;     int x ;     if ( i &gt; 0 )         asm mov x, 4     else         i = 7 ; }</pre> | <pre>int x() {     label:     -----     asm jmp label //Sprung zu label     ----- }</pre> |
|-------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|

## CPU-Register und Assemblerbefehle in Borland C++ 3.0

```
/* Beispiel.c Test CPU-Register und Assemblerbefehle in C */
// unsigned Pseudovariablen: _AX _AH _AL usw. ( Unterstrich und groß )
// Spezifizierer asm vor Assemblerzeile oder Assemblerblock
// keine Assembler-Kommentare mit ; sondern C-Kommentare mit //
// Beispiel: Schleife B-Port -> A-Port solange PC0 = High

#include <dos.h>           // für inp
#include "ports.h"         // Portdeklarationen
void main (void)
{
    unsigned char steuer = 0x8b ; // C-Variable mit Anfangswert lokal auf Stack

    _AL = steuer;           // Register <= C - Variable
    asm out SPORT, al       // Assembler-Zeile mit C-Kommentar

    while ( inp (CPORT) & 1) // keine Assembler-Ausdrücke ?
    {
        asm {               // Assembler-Block
            in al, BPORT;    // Semikolon am Ende
            out APORT, al    // oder Zeilenende cr
        }
    }
}
```

## Richtlinien für Interruptfunktionen in C

**Interruptfunktionen** sind far und erhalten den Spezifizierer **interrupt** bei der Prototyp-Deklaration und bei der **Definition** hinter **main**.

Sie übernehmen keine Parameter ( void ) und liefern kein Ergebnis / Rückgabewert ( void )

Beispiel: **void interrupt taste (void )**

. Übergabe von Daten zwischen main und der Interruptfunktion nur über globale Variablen, diese globalen Variablen **vor main** anlegen

## Interrupt-Vektortabelle

. Die Hauptfunktion main trägt die Adresse der Interruptfunktion mit der Funktion **poke** in die **Vektortabelle** ein. Man beachte den Abstand 0x100 vom Segmentanfang! ( .COM-Programme )

Eintragen von Interruptfunktionen in die Vektortabelle mit:

```
poke (0, vektor, FP_OFF (funktion) + 0x100);  
poke (0, vektor+2, FP_SEG (funktion));
```

Beispiel:

```
poke (0x0, IRQ7VEK, FP_OFF ( taste ) + 0x100 );           // Offset  
poke (0x0, IRQ7VEK+2, FP_SEG ( taste ) );                 // Segment
```

## Interrupt-Kontrolle

. Die Funktion **enable()** gibt den **IRQ-Interrupt frei** ( I-Bit = 1 , INTR-Freigabe ) und die Funktion **disable ()** sperrt den **IRQ-Interrupt** ( I-Bit = 0 )  
( **generelle Sperre/Freigabe** aller am INTR-Eingang über PIC hängender Interruptquellen )

. Freigabe und Sperre des Interrupts im Maskenregister des PIC durch die Hauptfunktion main mit inp und outp  
( **selektive Sperre/Freigabe** des jeweiligen Interrupts )

. **Bestätigung** des Interrupts im **Freigaberegister des PIC** muß durch die Interruptfunktion erfolgen.

## Beispiel für Interruptfunktionen in C ( für MVUS 8088, nicht PC )

### Programmbeispiel:

```

/*IntBeisp.c   Makro mit Parameter und Interrupt-Funktion   */
/* Taste PC7  gibt Zeichen aus String auf der Konsole aus   */

#include <dos.h>                                     // inp outp poke FP_OFF FP_SEG
#include "ports.h"                                   // Portdeklarationen
#define SBYTE 0x8b                                  // A=aus B=ein C=ein
#define IRQ7VEK 0x3c                                // Vektor 0x0f * 4 -> 0x3c IRQ7
#define IRQ7FRE 0x7f                                // 0111 1111 Freigabemaske
#define IRQ7BES 0x67                                // Bestaetigungscode fuer IRQ 7

// es folgt Makro mit Parameter zeichen
#define AUSGABE(zeichen) while(!(inp(SIOSTA) & 0x20)); outp(SIODAT,zeichen);

void interrupt taste (void);                        // Prototyp Interrupt-Funktion
unsigned char liste[13] = "ADAM und EVA";          // Stringkonstante global
unsigned int i;                                     // laufender Index global

void main(void)
{
    outp( SPORT, SBYTE );                          // PIO Richtung programmieren
    AUSGABE(10); AUSGABE(13);                      // neue Zeile
    i = 0 ;
    // Interrupt vorbereiten
    poke(0x0, IRQ7VEK, FP_OFF(taste) + 0x100 );    // Offset von taste eintragen
    poke(0x0, IRQ7VEK+2, FP_SEG(taste) );          // Segment eintragen
    outp(PICFREI, inp(PICFREI) & IRQ7FRE);          // Freigabe in PIC
    enable();                                       // I-Bit = 1: IRQ frei
    while(inp(CPORT) & 1);                        // Luafkontrolle mit PC0
    disable();                                     // I-Bit = 0: IRQ gesperrt
    outp(PICFREI, inp(PICFREI) | ~IRQ7FRE );       // IRQ7 in PIC sperren
}

void interrupt taste(void)                          // aufgerufen durch PC7-Taste
{
    unsigned char zeichen;                         // lokale Hilfsvariable
    zeichen = liste[i++];                         // Zeichen laden
    if (zeichen == 0) {i = 0; zeichen = liste[i++]; AUSGABE(10); AUSGABE(13);}
    AUSGABE(zeichen);                             // und ausgeben
    outp(PICBEST, IRQ7BES);                       // Interrupt bestaetigen
}

```